1 hph

FILED

2013 FEB 12 P 3: 26 (14)

RICHARD W. WIEKING
CLERK, U.S. DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

LSS

2

FAXED

Jeff D. Friedman (173886)
HAGENS BERMAN SOBOL SHAPIRO LLP
715 Hearst Avenue, Suite 202
Berkeley, CA 94710
Telephone: (510) 725-3000
Facsimile: (510) 725-3001
jefff@hbsslaw.com

Steve W. Berman (*Pro Hac Vice* Application to be filed)
Mark S. Carlson (*Pro Hac Vice* Application to be filed)
HAGENS BERMAN SOBOL SHAPIRO LLP
1918 Eighth Avenue, Suite 3300
Seattle, WA 98101
Telephone: (206) 623-7292
Facsimile: (206) 623-0594
steve@hbsslaw.com
markc@hbsslaw.com

Attorneys for Plaintiff
Brixham Solutions Ltd.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

JCS

BRIXHAM SOLUTIONS LTD., a British Virgin Islands International Business Company, Plaintiff, v. JUNIPER NETWORKS, INC., a Delaware corporation Defendants.	No. C 13 0616 COMPLAINT FOR PATENT INFRINGEMENT **DEMAND FOR JURY TRIAL**

Plaintiff Brixham Solutions, Ltd. ("Brixham"), for its complaint against defendant Juniper Networks, Inc., ("Juniper"), alleges as follows:

I. NATURE OF THE ACTION

1. This is an action under the patent laws of the United States of America, 35 U.S.C. §§ 1, *et seq.*, for infringement of patents assigned to Brixham.

II. THE PARTIES

2. Brixham is an international business company, organized and existing under the laws of the British Virgin Islands. Brixham maintains an office at OMC Chambers, Wickhams Cay 1, Road Town, Tortola, British Virgin Islands.

3. On information and belief, Juniper is a corporation organized and existing under the laws of the State of Delaware, with its principal place of business at 1194 North Mathilda Avenue, Sunnyvale, California 94089.

III. JURISDICTION AND VENUE

4. This Court has subject matter jurisdiction over this action pursuant to 28 U.S.C. §§ 1331 and 1338, in that this is a civil action for patent infringement arising under the Patent Laws of the United States, Title 35, United States Code.

5. Venue is proper in this judicial district under 28 U.S.C. §§ 1391(b)(1) and (c)(2), and 1400(b).

6. This Court has personal jurisdiction over Juniper because Juniper has its principal place of business in this judicial district, and Juniper conducts systematic and continuous business in California and within this judicial district, and has committed acts of infringement in California and within this judicial district.

IV. INTRADISTRICT ASSIGNMENT

7. This is an Intellectual Property Action to be assigned on a district-wide basis under Civil L.R. 3-2(c) and 3-5.

V. THE PATENTS-IN-SUIT

8. Brixham is the assignee and owner of U.S. Patent No. 7,535,895 (the "'895 Patent"), entitled *Selectively switching data between link interfaces and processing engines in a network switch*. The '895 Patent was duly and legally issued by the United States Patent and Trademark Office on May 19, 2009 and generally discloses and claims a network device for switching data from an incoming port to an outgoing port. The '895 Patent is attached to this Complaint as Exhibit A.

9. Brixham is the assignee and owner of U.S. Patent No. 7,609,621 (the "'621 Patent"), entitled *Automatic protection network switching*. The '621 Patent was duly and legally issued by the United States Patent and Trademark Office on October 27, 2009 and generally discloses and claims methods for connecting traffic from a service module to a first physical module having a link layer framer that is connected to a protected egress link. The '621 Patent is attached to this Complaint as Exhibit B.

10. Brixham is the assignee and owner of U.S. Patent No. 7,899,916 (the "'916 Patent"), entitled *Virtual service end-point*. The '916 Patent was duly and legally issued by the United States Patent and Trademark Office on March 1, 2011 and generally discloses and claims a network device to establish a connection between a first endpoint in a first protocol-specific network and a second endpoint in a second protocol-specific network. The '916 Patent is attached to this Complaint as Exhibit C.

11. Brixham is the assignee and owner of U.S. Patent No. 7,940,652 (the "'652 Patent"), entitled *Pseudowire protection using a standby pseudowire*. The '652 Patent was duly and legally issued by the United States Patent and Trademark Office on May 10, 2011 and generally discloses and claims systems and methods to provide protection to network traffic using one or more Pseudowires. The '652 Patent is attached to this Complaint as Exhibit D.

VI. BACKGROUND ON THE PATENTS-IN-SUIT AND JUNIPER

12. The technology disclosed and claimed in the patents-in-suit was developed by Hammerhead Systems, Inc. ("Hammerhead"). Hammerhead was a highly-regarded Silicon Valley

start-up founded in 2002 that liquidated in bankruptcy in 2009. A 2009 article published by the San Jose Mercury News described Hammerhead as "a case study of how even startups with talented teams, proven technology and lots of venture funding — about $110 million over seven years for Hammerhead — can be thwarted by circumstances beyond their control, including industry turmoil and the current global recession." One industry analyst quoted in the article noted with respect to Hammerhead that, "Sometimes the best technology does not win."

13. In its 2011 form 10k, Juniper states that, "We design, develop, and sell innovative products and services that together provide our customers with high-performance network infrastructure built on simplicity, security, openness and scale. We serve the high-performance networking requirements of global service providers, enterprises, and public sector organizations that view the network as critical to their success." Juniper also claims that "strong industry partnerships" enable it to "foster[] innovation across the network."

14. Juniper has incorporated Hammerhead's patented innovations into its own products as set forth below.

15. On information and belief, Juniper has actual knowledge of the asserted Brixham patents through their employment of former Hammerhead employees and service of this Complaint.

VII. <u>COUNT I</u>

INFRINGEMENT OF THE '895 PATENT

16. Brixham re-alleges and incorporates herein by reference the allegations stated in paragraphs 1-15 of this Complaint as if fully set forth herein.

17. Juniper has directly and/or indirectly infringed, and continues to directly and/or indirectly infringe (literally or under the doctrine of equivalents) one or more claims of the '895 Patent by making, using, offering for sale, selling and/or importing into the United States its M120 Multiservice Edge Router, and on information and belief other Juniper edge routers that incorporate the M120 technology including but not necessarily limited to the I-Chip, that either

alone, or in combination with other products for which they are intended to be used, include elements that meet all of the limitations of the infringed claims.

18. As a result of Juniper's infringement of the '895 Patent, Brixham has been and will continue to be irreparably harmed unless and until Juniper's infringement is enjoined by this Court.

19. As a result of Juniper's infringement of the '895 Patent, Brixham has been and will continue to be damaged in an amount to be proved at trial, but not less than a reasonable royalty for each infringement.

VIII. <u>COUNT II</u>

INFRINGEMENT OF THE '621 PATENT

20. Brixham re-alleges and incorporates herein by reference the allegations stated in paragraphs 1-19 of this Complaint as if fully set forth herein.

21. Juniper has directly and/or indirectly infringed, and continues to directly and/or indirectly infringe (literally or under the doctrine of equivalents) one or more claims of the '621 Patent by making, using, offering for sale, selling and/or importing into the United States its M120 Multiservice Edge Router, and on information and belief other Juniper edge routers that incorporate the M120 technology including but not necessarily limited to the I-Chip, that either alone, or in combination with other products for which they are intended to be used, include elements that meet all of the limitations of the infringed claims.

22. As a result of Juniper's infringement of the '621 Patent, Brixham has been and will continue to be irreparably harmed unless and until Juniper's infringement is enjoined by this Court.

23. As a result of Juniper's infringement of the '621 Patent, Brixham has been and will continue to be damaged in an amount to be proved at trial, but not less than a reasonable royalty for each infringement.

IX. <u>COUNT III</u>

INFRINGEMENT OF THE '916 PATENT

24. Brixham re-alleges and incorporates herein by reference the allegations stated in paragraphs 1-23 of this Complaint as if fully set forth herein.

25. Juniper has been and continues to directly and/or indirectly infringe (literally or under the doctrine of equivalents) one or more claims of the '916 Patent by making, using, offering for sale, selling and/or importing into the United States products that either alone, or in combination with other products for which they are intended to be used, include elements that meet all of the limitations of the infringed claims. Accused M120 Multiservice Edge Routers, and on information and belief other Juniper edge routers incorporating the Junos network operating system (version 10.4 and later) with Translational Cross-Connect for creating virtual switches and Layer 2.5 VPNs (Virtual Private Networks) and Layer 2.5 circuits.

26. As a result of Juniper's infringement of the '916 Patent, Brixham has been and will continue to be irreparably harmed unless and until Juniper's infringement is enjoined by this Court.

27. As a result of Juniper's infringement of the '916 Patent, Brixham has been and will continue to be damaged in an amount to be proved at trial, but not less than a reasonable royalty for each infringement.

X. COUNT IV

INFRINGEMENT OF THE '652 PATENT

28. Brixham re-alleges and incorporates herein by reference the allegations stated in paragraphs 1-27 of this Complaint as if fully set forth herein.

29. Juniper has been and continues to directly and/or indirectly infringe (literally or under the doctrine of equivalents) one or more claims of the '652 Patent by making, using, offering for sale, selling and/or importing into the United States products that either alone, or in combination with other products for which they are intended to be used, include elements that meet all of the limitations of the infringed claims. Accused M120 Multiservice Edge Routers, and on information and belief other Juniper edge routers, that employ Junos network operating system redundant Pseudowires that establish Multi-segment Layer 2.5 Pseudowire switching.

30. As a result of Juniper's infringement of the '652 Patent, Brixham has been and will continue to be irreparably harmed unless and until Juniper's infringement is enjoined by this Court.

31. As a result of Juniper's infringement of the '652 Patent, Brixham has been and will continue to be damaged in an amount to be proved at trial, but not less than a reasonable royalty for each infringement.

XI. PRAYER FOR RELIEF

Wherefore, Brixham respectfully requests that this Court:

1. Enter a judgment in favor of Brixham that Juniper has infringed one or more claims of the '895 Patent, '621 Patent, '916 Patent, and '652 Patent;

2. Grant a permanent injunction enjoining Juniper, its officers, directors, agents, servants, affiliates, employees, successors, assigns, divisions, branches, subsidiaries, parents and all others acting in active concert therewith from infringing and/or inducing others to infringe or contribute to the infringement of the '895 Patent, '621 Patent, '916 Patent, and '652 Patent;

3. Award Brixham damages in an amount sufficient to compensate for Juniper's infringement of the '895 Patent, '621 Patent, '916 Patent, and '652 Patent, in an amount to be proved at trial, but not less than a reasonable royalty;

4. Award prejudgment interest to Brixham under 35 U.S.C. § 284;

5. If supported by the evidence, declare this case exceptional under 35 U.S.C. § 285 and award Brixham reasonable attorney's fees; and

6. Grant Brixham such other and further relief as this Court deems just and equitable.

//

//

//

//

//

//

//

//

XII. <u>DEMAND FOR JURY TRIAL</u>

Brixham hereby demands a trial by jury on all issues so triable.

DATED: February 12, 2013

HAGENS BERMAN SOBOL SHAPIRO LLP

By: [illegible]
Jeff D. Friedman (173886)

715 Hearst Avenue, Suite 202
Berkeley, CA 94710
Telephone: (510) 725-3000
Facsimile: (510) 725-3001
jefff@hbsslaw.com

Steve W. Berman (*Pro Hac Vice* Application to be Filed)
Mark S. Carlson (*Pro Hac Vice* Application to be Filed)
HAGENS BERMAN SOBOL SHAPIRO LLP
1918 Eighth Avenue, Suite 3300
Seattle, WA 98101
Telephone: (206) 623-7292
Facsimile: (206) 623-0594
steve@hbsslaw.com
markc@hbsslaw.com

Attorneys for Plaintiff
Brixham Solutions Ltd.

Exhibit A

US007535895B2

(12) **United States Patent**
Medved et al.

(10) **Patent No.: US 7,535,895 B2**
(45) **Date of Patent: May 19, 2009**

(54) SELECTIVELY SWITCHING DATA BETWEEN LINK INTERFACES AND PROCESSING ENGINES IN A NETWORK SWITCH

(75) Inventors: **Jan Medved**, Pleasanton, CA (US); **Alex Dadnam**, San Ramon, CA (US); **Sameer Kanagala**, San Carlos, CA (US); **Fong Liaw**, Cupertino, CA (US); **John Burns**, Los Altos, CA (US); **David Bumstead**, San Jose, CA (US)

(73) Assignee: **Hammerhead Systems, Inc.**, Mountain View, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 925 days.

(21) Appl. No.: **10/447,825**

(22) Filed: **May 29, 2003**

(65) **Prior Publication Data**

US 2004/0240470 A1 Dec. 2, 2004

(51) **Int. Cl.**
H04J 3/16 (2006.01)

(52) **U.S. Cl.** **370/360**; 370/376; 370/467; 370/469

(58) **Field of Classification Search** 370/469, 370/466, 471, 376, 907, 395.51
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,521,919	A *	5/1996	Anderson et al.	370/376
5,528,587	A	6/1996	Galand et al.	370/412
5,712,853	A *	1/1998	Mathur et al.	370/467
6,332,198	B1	12/2001	Simons et al.	714/6
6,519,257	B1	2/2003	Brueckheimer et al.	
6,891,836	B1 *	5/2005	Chen et al.	370/395.51
6,944,153	B1 *	9/2005	Buckland et al.	370/376
6,973,028	B1 *	12/2005	Huai et al.	370/222
7,130,276	B2	10/2006	Chen et al.	370/249
7,184,440	B1 *	2/2007	Sterne et al.	370/395.52
2002/0049608	A1	4/2002	Hartsell et al.	

(Continued)

FOREIGN PATENT DOCUMENTS

WO	WO01/90843	11/2001

OTHER PUBLICATIONS

Patridge et al., A 50-Gb/s IP Router, IEEE, 12 pages, 1998.
James Aweya, IP Router Architectures: An Overview, Nortel Networks, 48 pages, 1999.

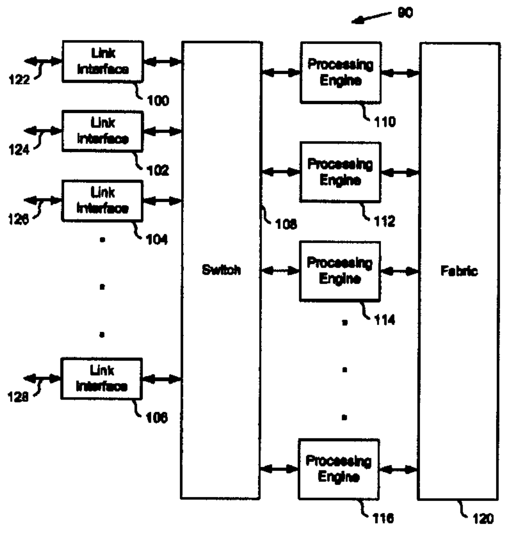
(Continued)

Primary Examiner—Chi H. Pham
Assistant Examiner—Shick Hom
(74) *Attorney, Agent, or Firm*—Van Pelt, Yi & James LLP

(57) **ABSTRACT**

Technology is disclosed for directing data through a network switch. One version of a network switch employs a mid-plane architecture that allows data to be directed between any link interface and any processing engine. Each time slot of data from an ingress link interface can be separately directed to any ingress processing engine. Each time slot of data from an egress processing engine can be separately directed to any egress link interface that supports the lower level protocol for the data. In one version of the switch, each processing engine in the network switch has the ability to service all of the protocols from the layers of the OSI model that are supported by the switch and not handled on the link interfaces. This allows the switch to allocate processing engine resources, regardless of the protocols employed in the data passing through the switch.

38 Claims, 13 Drawing Sheets



U.S. PATENT DOCUMENTS

2002/0116485	A1	8/2002	Black et al.	709/223
2003/0236919	A1	12/2003	Johnson et al.	
2004/0004961	A1	1/2004	Lakshmanamurthy et al.	
2004/0240470	A1	12/2004	Medved et al.	370/907
2007/0280223	A1*	12/2007	Pan et al.	370/360

OTHER PUBLICATIONS

Kumagai et al., IP Router for Next-Generation Network, Fujitsu Sci. Tech, 11 pages, 2001.

Niraj Shah, Understanding Network Processor, Berkeley University, 93 pages, 2001.

* cited by examiner

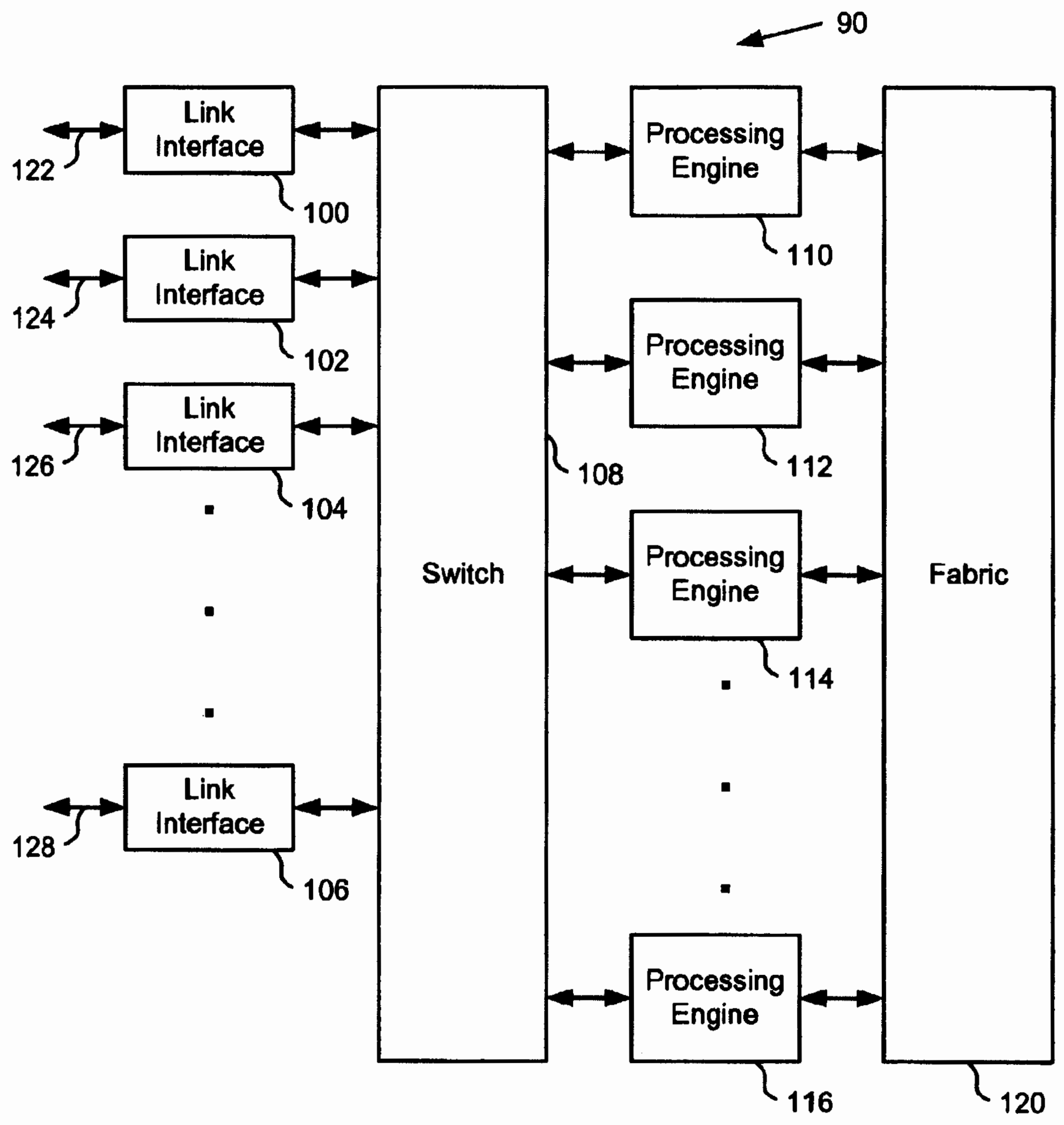

90
Link Interface
Link Interface
Link Interface
Link Interface
122
124
126
128
100
102
104
106
Switch
108
Processing Engine
Processing Engine
Processing Engine
Processing Engine
110
112
114
116
Fabric
120

FIG. 1

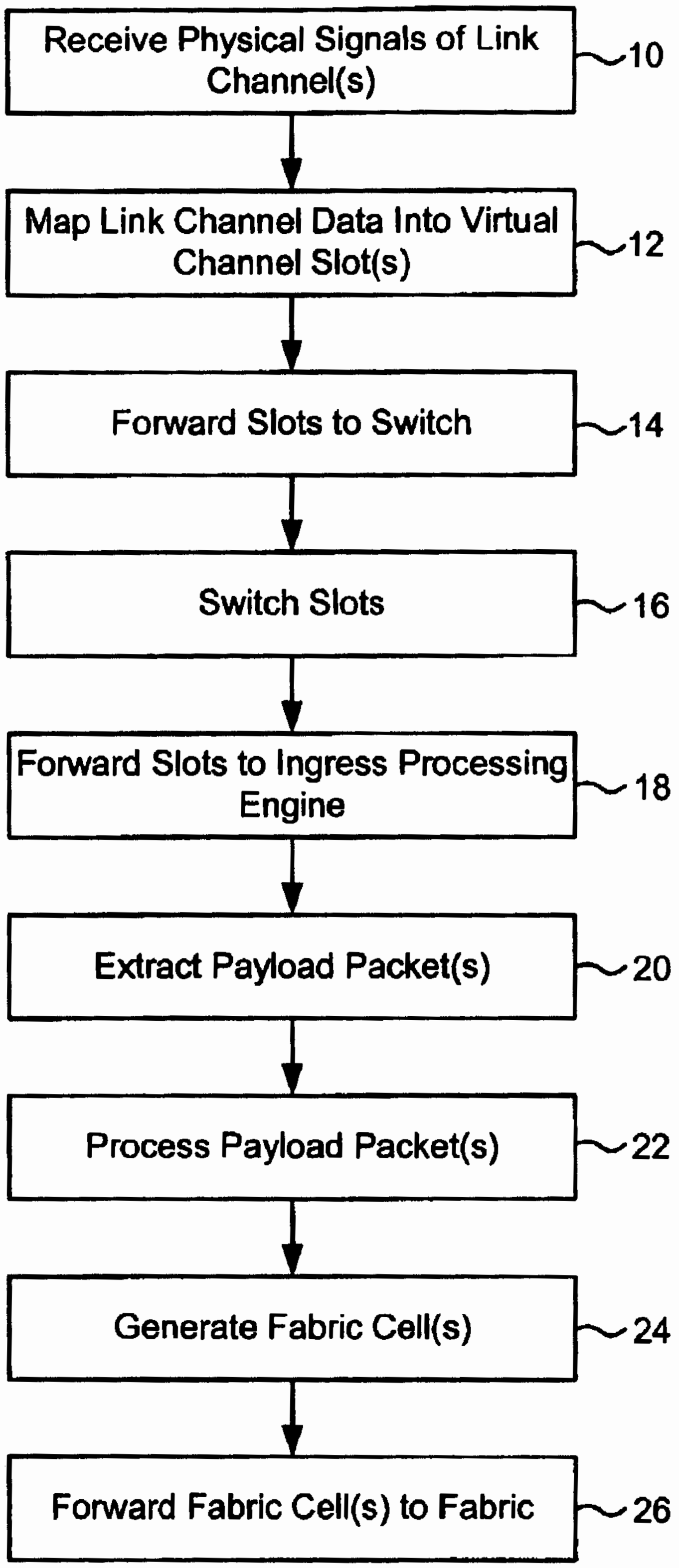

Receive Physical Signals of Link Channel(s)
10
Map Link Channel Data Into Virtual Channel Slot(s)
12
Forward Slots to Switch
14
Switch Slots
16
Forward Slots to Ingress Processing Engine
18
Extract Payload Packet(s)
20
Process Payload Packet(s)
22
Generate Fabric Cell(s)
24
Forward Fabric Cell(s) to Fabric
26

FIG. 2A

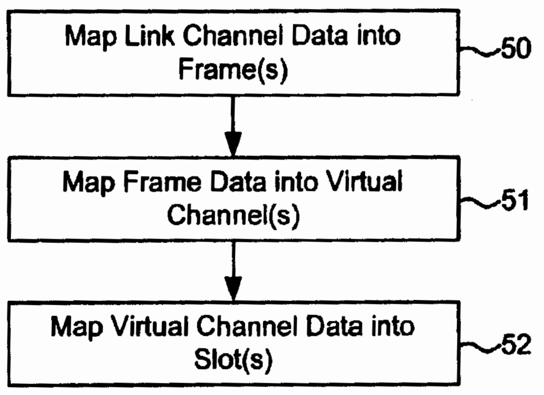

Map Link Channel Data into Frame(s)
50
Map Frame Data into Virtual Channel(s)
51
Map Virtual Channel Data into Slot(s)
52

FIG. 3A

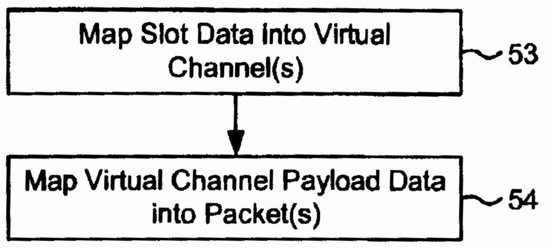

Map Slot Data into Virtual Channel(s)
53
Map Virtual Channel Payload Data into Packet(s)
54

FIG. 3B

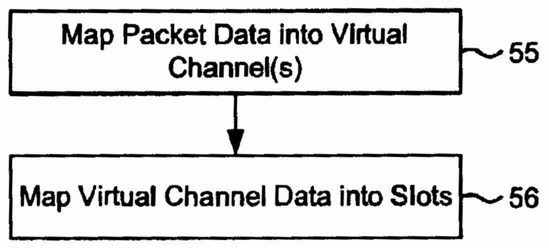

Map Packet Data into Virtual Channel(s)
55
Map Virtual Channel Data into Slots
56

FIG. 3C

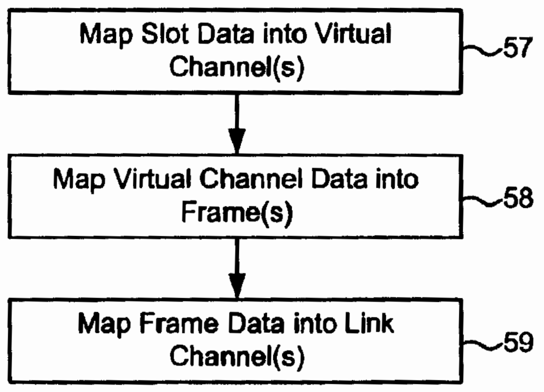

Map Slot Data into Virtual Channel(s)
57
Map Virtual Channel Data into Frame(s)
58
Map Frame Data into Link Channel(s)
59

FIG. 3D

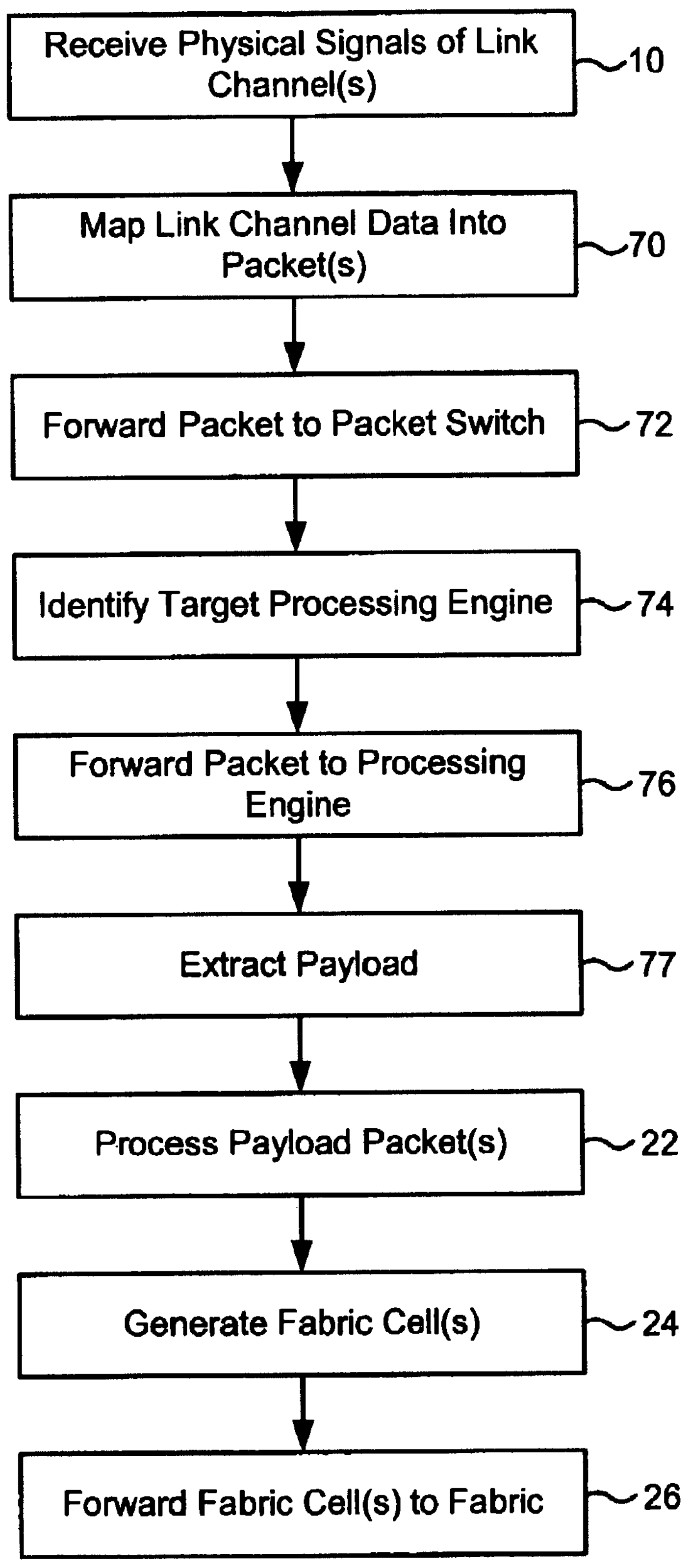
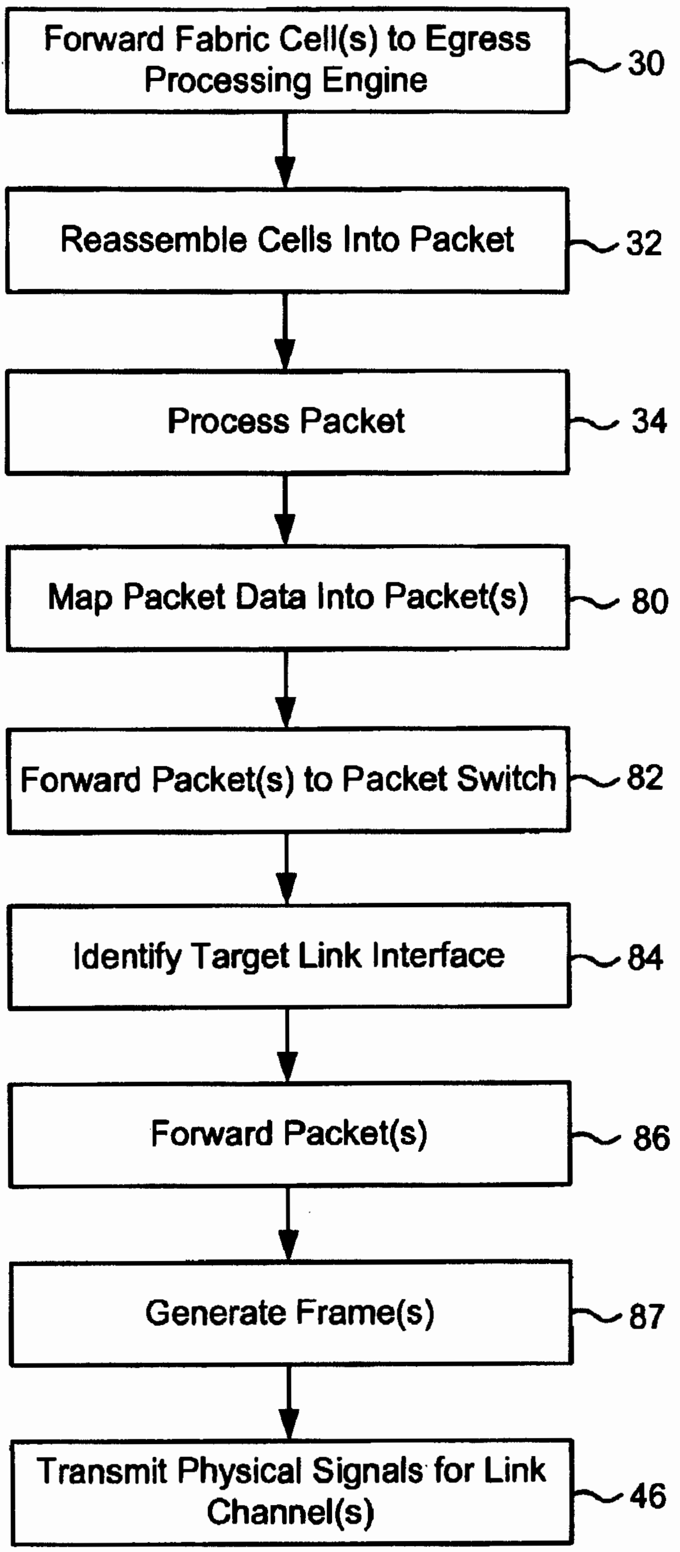

Receive Physical Signals of Link Channel(s)
10
Map Link Channel Data Into Packet(s)
70
Forward Packet to Packet Switch
72
Identify Target Processing Engine
74
Forward Packet to Processing Engine
76
Extract Payload
77
Process Payload Packet(s)
22
Generate Fabric Cell(s)
24
Forward Fabric Cell(s) to Fabric
26

FIG. 6



Forward Fabric Cell(s) to Egress Processing Engine
30
Reassemble Cells Into Packet
32
Process Packet
34
Map Packet Data Into Packet(s)
80
Forward Packet(s) to Packet Switch
82
Identify Target Link Interface
84
Forward Packet(s)
86
Generate Frame(s)
87
Transmit Physical Signals for Link Channel(s)
46

FIG. 7

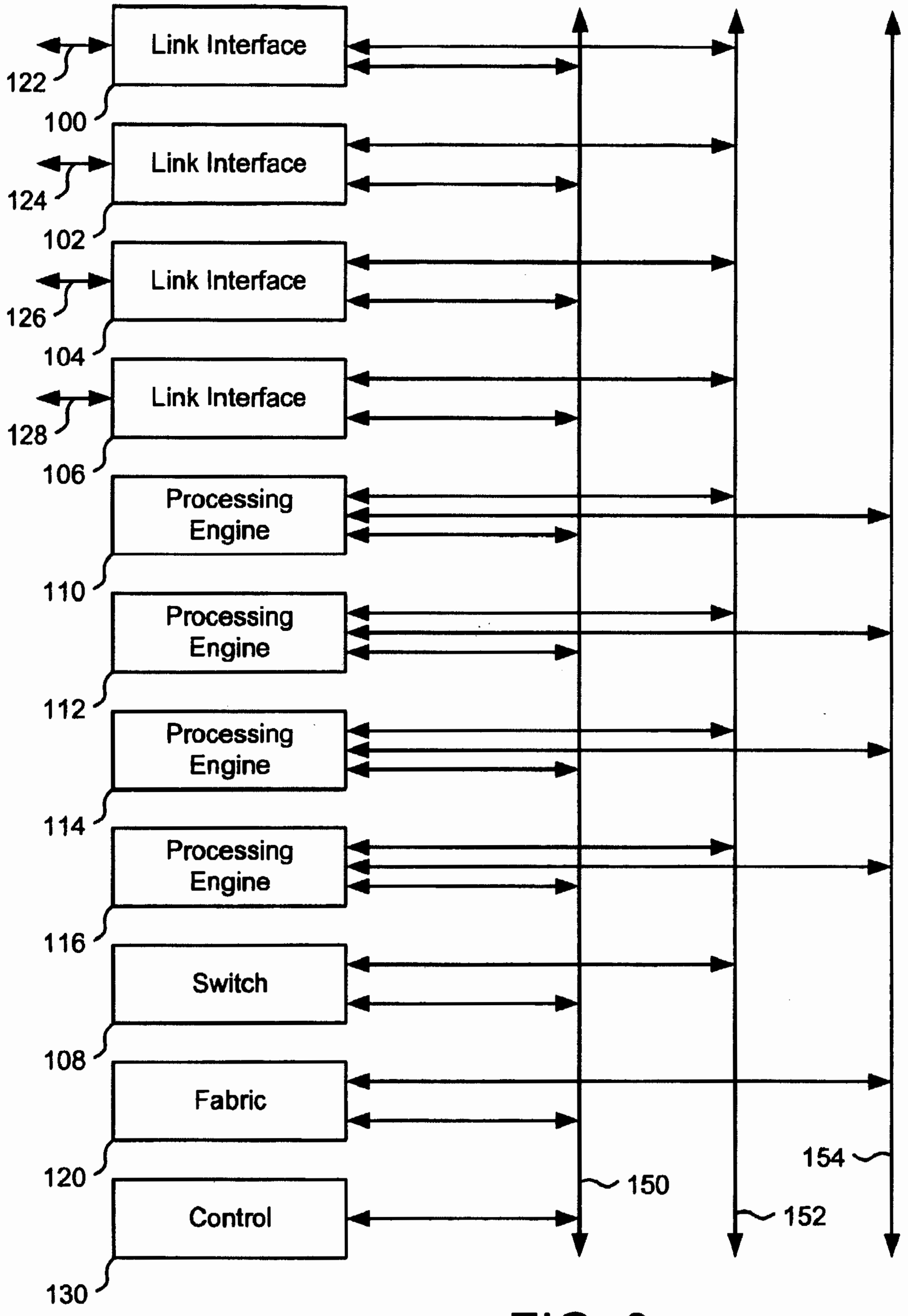

Link Interface
122
100
Link Interface
124
102
Link Interface
126
104
Link Interface
128
106
Processing Engine
110
Processing Engine
112
Processing Engine
114
Processing Engine
116
Switch
108
Fabric
120
Control
130
150
152
154

FIG. 8

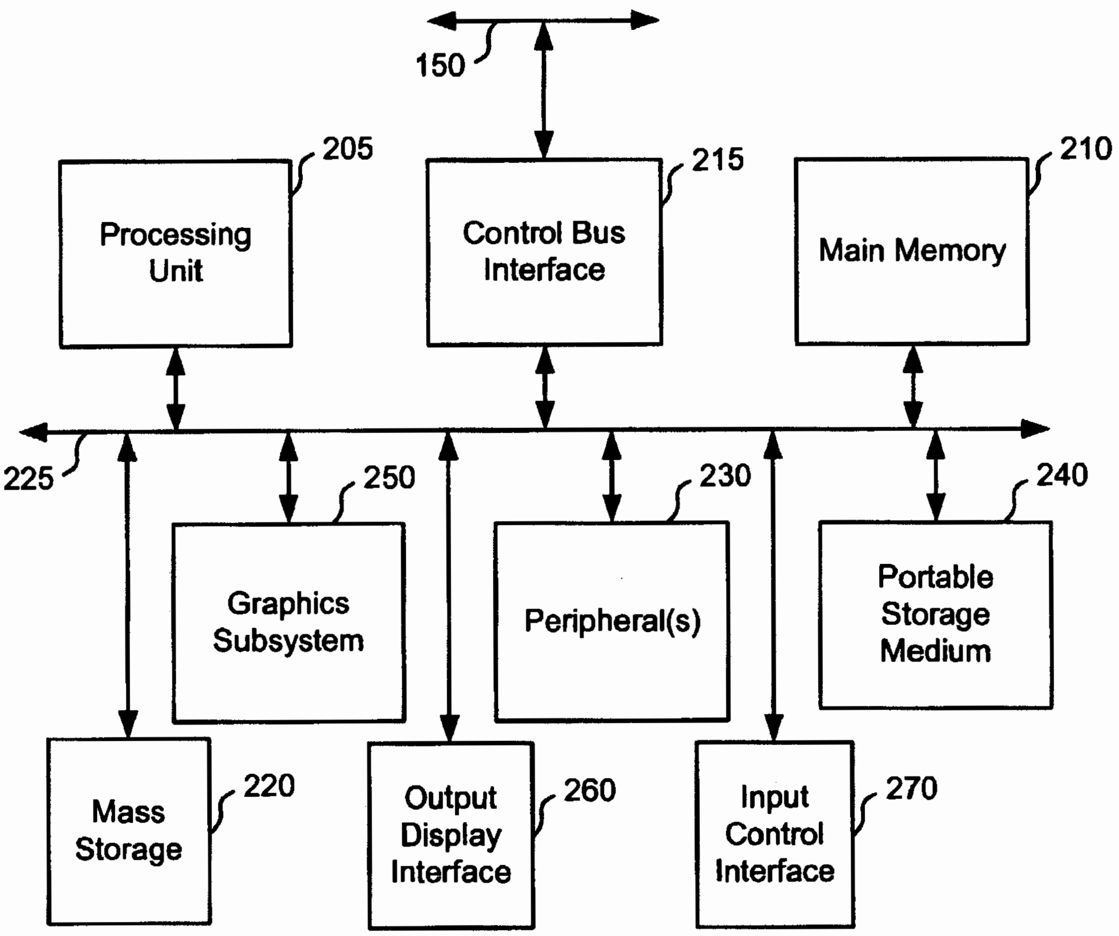

150
Processing Unit
205
Control Bus Interface
215
Main Memory
210
225
Graphics Subsystem
250
Peripheral(s)
230
Portable Storage Medium
240
Mass Storage
220
Output Display Interface
260
Input Control Interface
270

FIG. 9

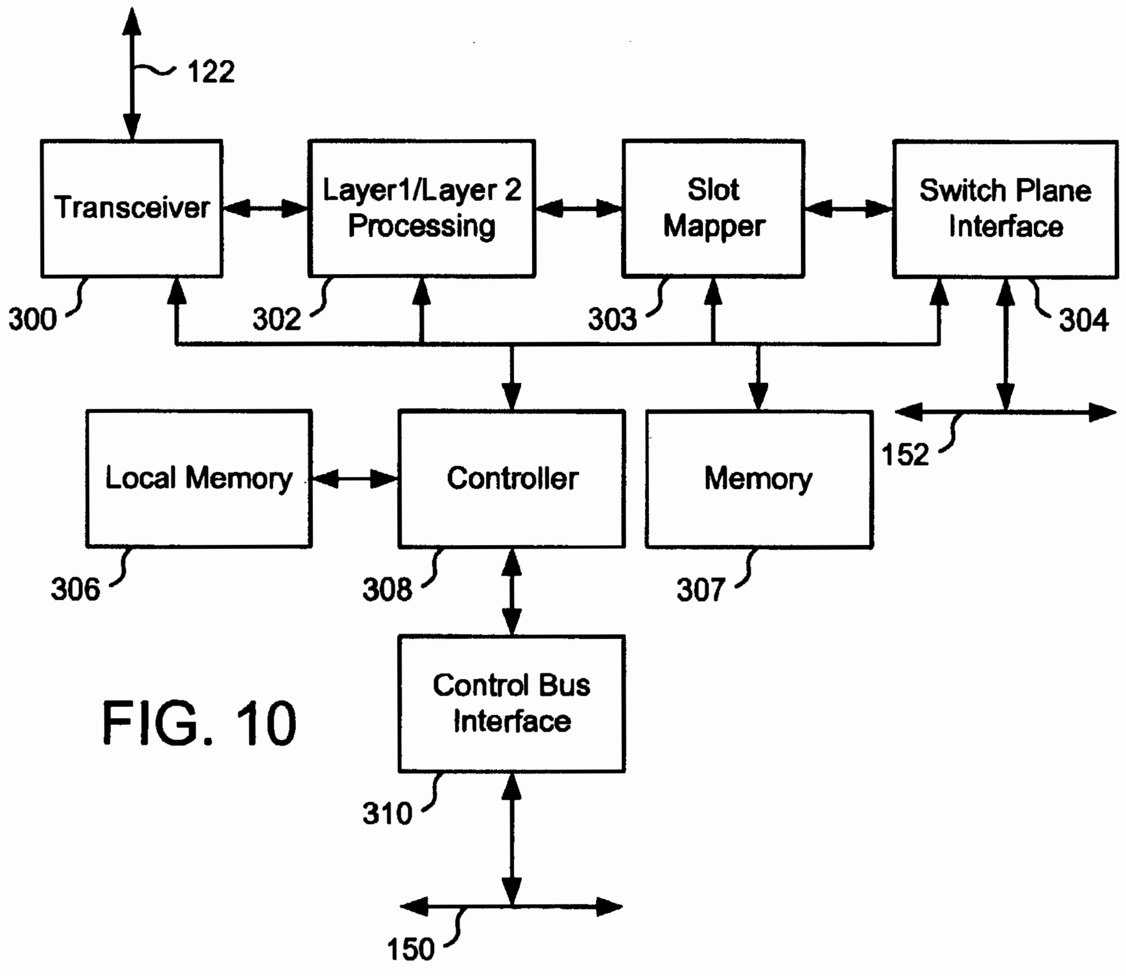

122
Transceiver
Layer1/Layer 2 Processing
Slot Mapper
Switch Plane Interface
300
302
303
304
Local Memory
Controller
Memory
152
306
308
307
Control Bus Interface
310
150

FIG. 10

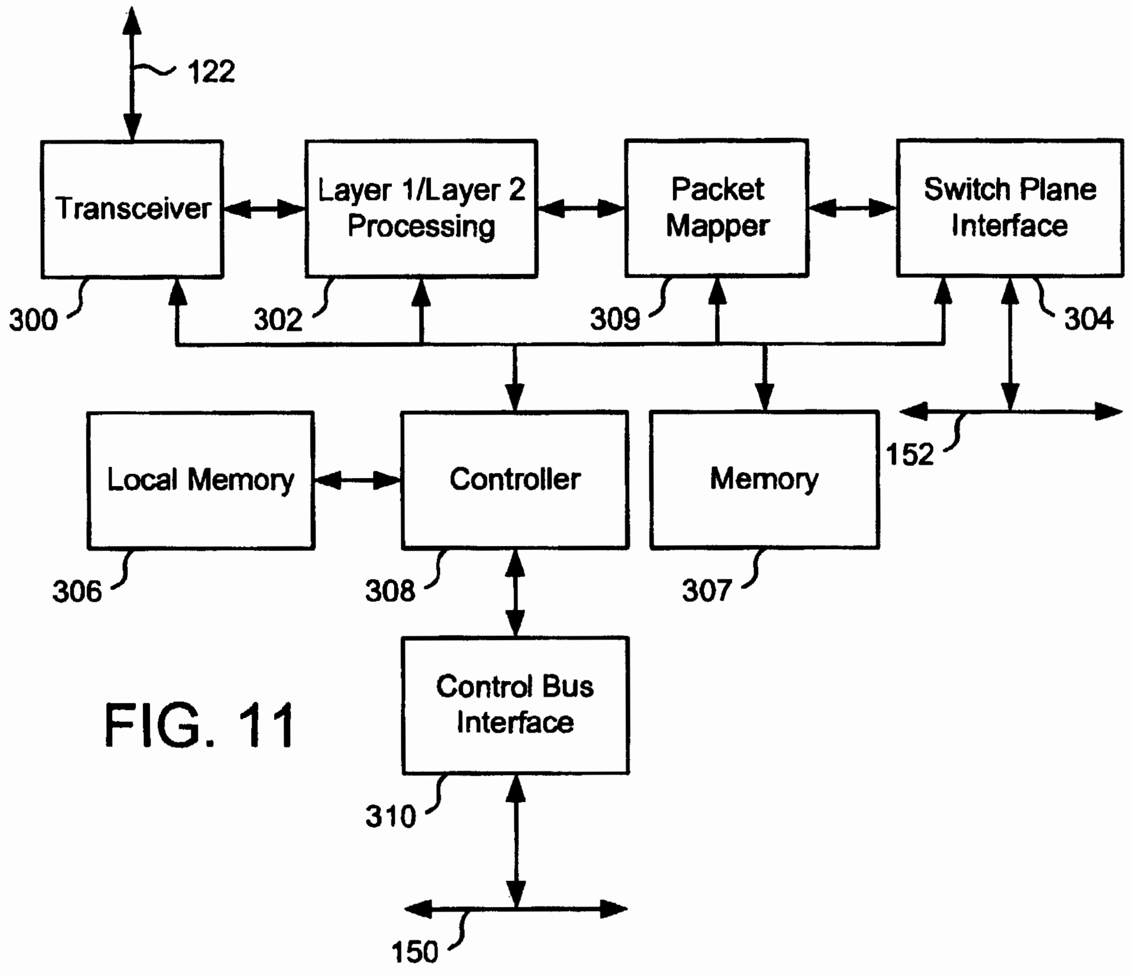

122
Transceiver
Layer 1/Layer 2 Processing
Packet Mapper
Switch Plane Interface
300
302
309
304
Local Memory
Controller
Memory
152
306
308
307
Control Bus Interface
310
150

FIG. 11

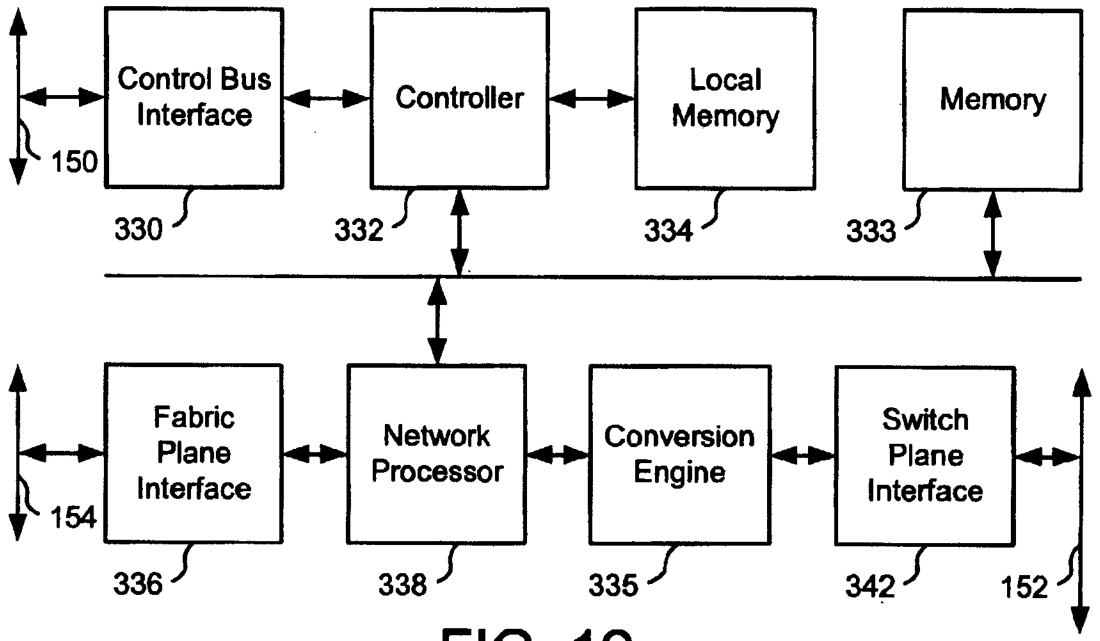

Control Bus Interface
Controller
Local Memory
Memory
150
330
332
334
333
Fabric Plane Interface
Network Processor
Conversion Engine
Switch Plane Interface
154
336
338
335
342
152

FIG. 12

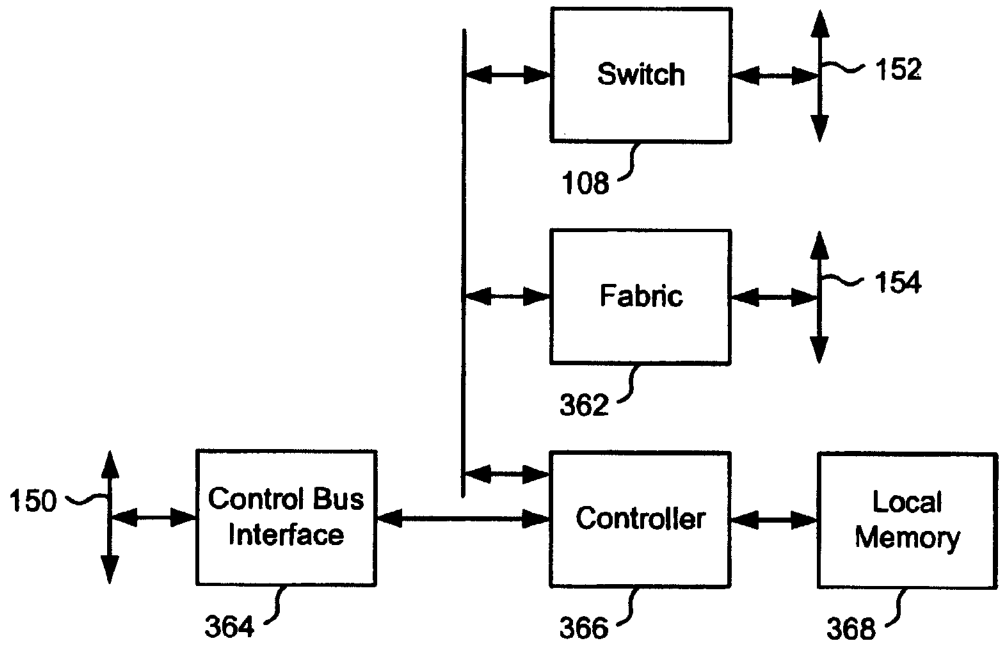

Switch
152
108
Fabric
154
362
150
Control Bus Interface
Controller
Local Memory
364
366
368

FIG. 13

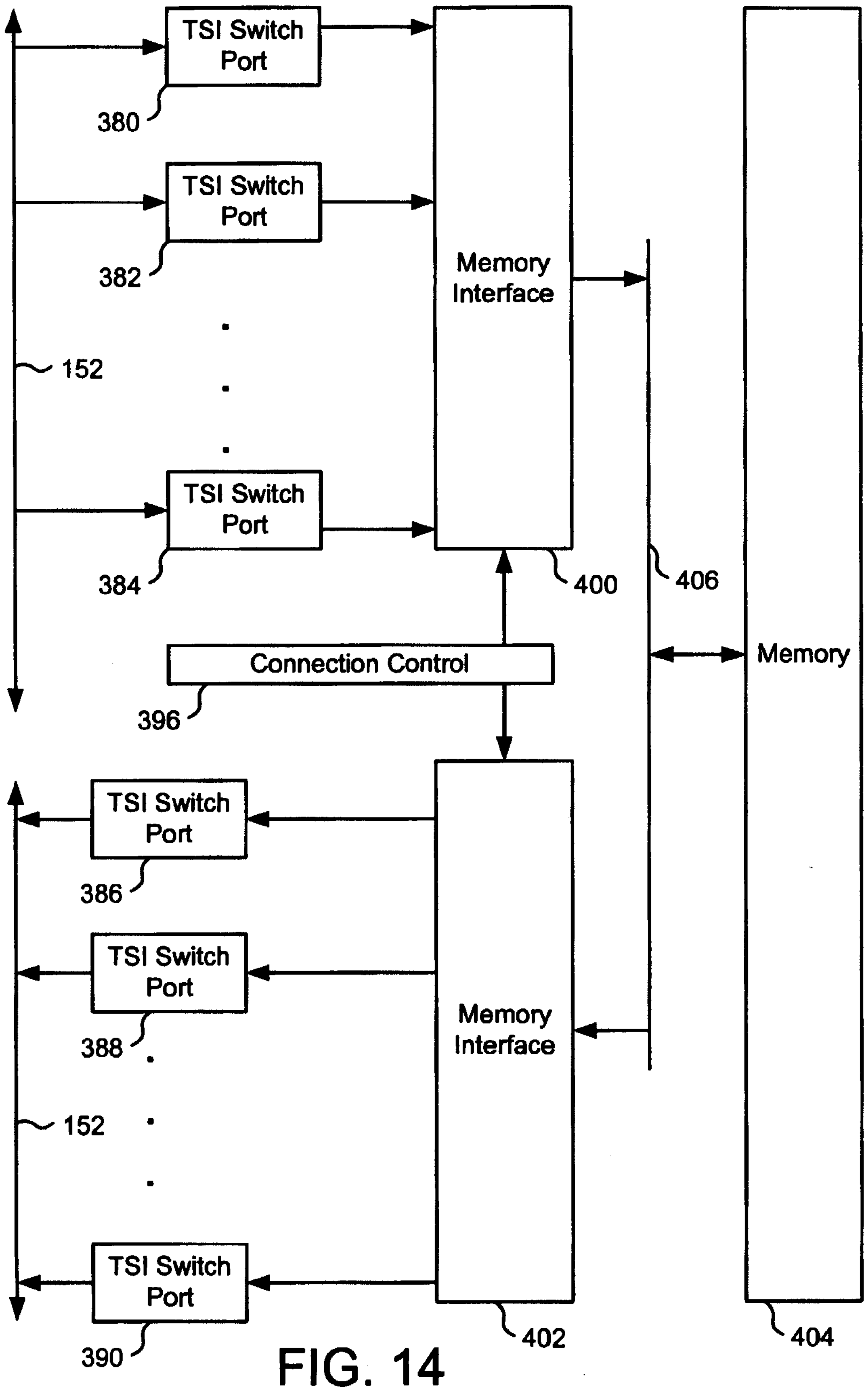

TSI Switch Port
380
TSI Switch Port
382
152
TSI Switch Port
384
Memory Interface
400
406
Memory
Connection Control
396
TSI Switch Port
386
TSI Switch Port
388
152
Memory Interface
TSI Switch Port
390
402
404

FIG. 14

SELECTIVELY SWITCHING DATA BETWEEN LINK INTERFACES AND PROCESSING ENGINES IN A NETWORK SWITCH

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention is directed to network switching technology.

2. Description of the Related Art

Network switches process data from an incoming port and direct it to an outgoing port. Network switches offer a variety of services, including support for virtual private networks ("VPNs"). The increased popularity of the Internet and other network related technologies has increased performance demands on network switches. Network switches need to efficiently manage their resources, while supporting multiple physical signaling standards and higher-level protocols.

A typical network switch includes link interfaces for exchanging data with physical signaling mediums. The mediums carry data according to multiple physical signaling protocols. Each link interface supports a physical signaling standard from the physical layer (Layer 1) of the Open Systems Interconnection ("OSI") model. In one example, a link interface supports a network connection for Synchronous Optical Network ("SONET") at Optical Carrier level **48** ("OC-48"). In another example, a link interface supports the physical layer of Gigabit Ethernet. Some link interfaces also support portions of the data-link layer (Layer 2) in the OSI model, such as the Media Access Control ("MAC") of Gigabit Ethernet.

Incoming data also conforms to one or more higher-level protocols, such as High-level Data Link Control ("HDLC"), Point-to-Point Protocol ("PPP"), Frame Relay, Asynchronous Transfer Mode ("ATM"), and other protocols in higher layers of the OSI model. Each link interface forwards incoming data to a processing engine in the network switch that supports a higher-level protocol for the data—a network switch may have one or multiple processing engines. The processing engine interprets the data and performs any desired processing, such as packet processing according to one or more layers in the OSI model. A variety of different processing operations can be performed, based on the services supported in the network switch.

The processing engine identifies an egress link interface for incoming data and arranges for the data to be forwarded to the egress link interface. The egress link interface delivers the data to a desired network medium connection. In network switches with multiple processing engines, an ingress processing engine directs the data through an egress processing engine for forwarding to the egress link interface.

Network switches can be implemented as either single card or multiple card systems. A single card system implements all of the switch's link interfaces and processing engines on a single printed circuit board ("PCB"). These types of systems are typically simple switches with the link interfaces being directly connected to a single processing engine. If multiple processing engines are employed, each processing engine is typically connected to a fixed set of link interfaces. The processing engines are also coupled to a fabric or interconnect mesh for exchanging data with each other. The connections between a set of link interfaces and a processing engine cannot be altered, regardless of the level of traffic through the link interfaces. This can result in an inefficient use of system resources—one processing engine can be over utilized, while another processing engine is under utilized. The fixed relationship between link interfaces and processing engines does not permit incoming link interface traffic to be redirected to a processing engine with excess resources.

Multiple card network switches implement link interface functionality and processing engine functionality on multiple PCBs—allowing the network switch to support more link interfaces and processing engines than a single card switch. Two types of multiple card systems are the backplane architecture and the mid-plane architecture.

In the backplane architecture, multiple line cards are coupled together over a backplane. A line card includes a processing engine coupled to one or more link interfaces for interfacing to physical mediums. Multiple line cards are coupled to the backplane. Incoming data is switched from an ingress line card to an egress line card for transmission onto a desired network medium. One type of backplane architecture includes a fabric card coupled to the backplane for facilitating the transfer of data between line cards.

The backplane architecture also fails to efficiently allocate system resources. The relationship between link interfaces and processing engines is fixed. Each processing engine resides on a line card with its associated link interfaces. The processing engine can only communicate with link interfaces on its line card. Incoming traffic through the link interfaces on one card can be very heavy. None of this traffic can be directed to a processing engine on another line card for ingress processing. This is wasteful if processing engines on other cards have excess resources available.

In the mid-plane architecture, multiple cards are coupled to a mid-plane that facilitates communication between the cards. The switch includes processing engine cards and link interface cards. Each processing engine card includes a processing engine, and each link interface card has one or more link interfaces. One type of link interface typically performs processing on data at Layer 1 of the OSI model. Each processing engine card processes data at Layer 2 in the OSI model and higher. A processing engine card provides only one type of Layer 2 processing—requiring the switch to contain at least one processing engine card for each type of Layer 2 protocol supported in the switch. In some instances, link interfaces perform a portion of the Layer 2 processing, such as Gigabit Ethernet MAC framing. Another type of link interface only performs a portion of the Layer 1 processing—operating only as a transceiver. In this implementation, the processing engine card also performs the remaining portion of the Layer 1 processing. Each processing engine card only supports one type of Layer 1 protocol—requiring the switch to contain at least one processing engine card for each type of Layer 1 protocol supported in the switch.

In operation, an ingress link interface card forwards incoming data to an ingress processing engine card. The ingress processing engine card is dedicated to processing data according to the higher-level protocol employed in the incoming data. The ingress processing engine passes the processed incoming data to an egress processing engine card for further processing and forwarding to an egress link interface card. In one implementation, the switch includes a fabric card for switching data from one processing engine card to another processing engine card.

One type of mid-plane switch passes data between link interface cards and processing engine cards over fixed traces in the mid-plane. This creates the same inefficient use of system resources explained above for the single card architecture and backplane architecture.

Another type of mid-plane switch employs a programmable connection card to direct data between link interface cards and processing engine cards. Each link interface card has one or more serial outputs for ingress data coupled to the connection card and one or more serial inputs for egress data coupled to the connection card. Each processing engine card has one or more serial outputs for egress data coupled to the connection card and one or more serial inputs for ingress data coupled to the connection card. For each link interface card output, the connection card forms a connection with one processing engine card input for transferring data. For each processing engine card output, the connection card forms a connection with one link interface card input for transferring data. Data from one card output cannot be directed to multiple card inputs through the connection card.

Even with the connection card, the mid-plane architecture wastes resources. Protocol specific processing engine cards can be wasteful. If the network switch receives a disproportionately large percentage of data according to one protocol, the processing engine supporting that protocol is likely to be over utilized. Meanwhile, processing engine cards for other protocols remain under utilized with idle processing bandwidth. The resource inefficiency of existing mid-plane systems is worse when a switch includes redundant processing engine cards. The network switch requires at least one redundant processing engine card for each protocol supported in the network switch, regardless of the protocol's utilization level.

SUMMARY OF THE INVENTION

The present invention, roughly described, pertains to technology for efficiently utilizing resources within a network switch. One implementation of a network switch employs a mid-plane architecture that allows data to be directed between any link interface and any processing engine. In one implementation, each link interface can have a single data stream or a channelized data stream. Each channel of data from a link interface can be separately directed to any processing engine. Similarly, each channel of data from a processing engine can be separately directed to any link interface. In one embodiment, each processing engine in the network switch has the ability to service all of the protocols from the layers of the OSI model that are supported by the switch and not handled on the link interfaces. This allows the switch to allocate processing engine resources, regardless of the protocols employed in the data passing through the switch.

One embodiment of the network switch includes link interfaces, processing engines, a switched fabric between the processing engines, and a switch between the link interfaces and processing engines. In one implementation, the switch between the link interfaces and processing engines is a time slot interchange ("TSI") switch. An ingress link interface receives incoming data from a physical signaling medium. The ingress link interface forwards incoming data to the TSI switch. The TSI switch directs the data to one or more ingress processing engines for processing, such as forwarding at the Layer 2 or Layer 3 level of the OSI model. In one implementation, the TSI switch performs Time Division Multiplexing ("TDM") switching on data received from each link interface—separately directing each time slot of incoming data to the proper ingress processing engine. In an alternate embodiment, the TSI switch is replaced by a packet switch. The information exchanged between link interfaces and processing engines is packetized and switched through the packet switch.

The ingress processing engine sends data to the packet switch fabric, which directs packets from the ingress processing engine to one or more egress processing engines for further processing and forwarding to the TSI switch. The TSI

switch directs the data to one or more egress link interfaces for transmission onto a physical medium. One implementation of the TSI switch performs TDM switching on data streams received from each processing engine—separately directing each time slot of incoming data to the proper egress link interface. In an alternate embodiment, the TSI switch is replaced by a packet switch that performs packet switching.

The switch between the link interfaces and processing engines can be any multiplexing switch—a switch that multiplexes data from multiple input interfaces onto a single output interface and demultiplexes data from a single input interface to multiple output interfaces. The above-described TSI switch and packet switch are examples of a multiplexing switch.

In one example, the TSI switch receives data from link interfaces and processing engines in the form of SONET STS-48 frames. The TSI switch has the ability to switch time slots in the SONET frame down to the granularity of a single Synchronous Transport Signal—1 ("STS-1") channel. In alternate embodiments, the TSI switch can switch data at a higher or lower granularity. Further implementations of the TSI switch perform virtual concatenation—switching time slots for multiple STS-1 channels that operate together as a higher throughput virtual channel, such as a STS-3 channel.

The operation of the TSI switch and protocol independence of the processing engines facilitates bandwidth pooling within the network switch. When a processing engine becomes over utilized, a channel currently supported by the processing engine can be diverted to any processing engine that is not operating at full capacity. This redirection of network traffic can be performed at the STS-1 channel level or higher. Similar adjustments can be made when a processing engine or link interface are under utilized. Bandwidth pooling adjustments can be made when the network switch is initialized and during the switch's operation. The network switch also provides efficient redundancy—a single processing engine can provide redundancy for many other processing engines, regardless of the protocols embodied in the underlying data. Any processing engine can be connected to any channel on any link interface—allowing any processing engine in the network switch to back up any other processing engine in the switch. This easily facilitates the implementation of 1:1 or 1:N processing engine redundancy. In one implementation, the efficient distribution of resources allows for a 2:1 ratio of link interfaces to processing engines, so that each link interface has redundancy and no processing engine is required to sit idle.

The present invention can be accomplished using hardware, software, or a combination of both hardware and software. The software used for the present invention is stored on one or more processor readable storage media including hard disk drives, CD-ROMs, DVDs, optical disks, floppy disks, tape drives, RAM, ROM or other suitable storage devices. In alternative embodiments, some or all of the software can be replaced by dedicated hardware including custom integrated circuits, gate arrays, FPGAs, PLDs, and special purpose computers. In one embodiment, software implementing the present invention is used to program one or more processors, including microcontrollers and other programmable logic. The processors can be in communication with one or more storage devices, peripherals and/or communication interfaces.

These and other objects and advantages of the present invention will appear more clearly from the following description in which the preferred embodiment of the invention has been set forth in conjunction with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram depicting one embodiment of a network switch in accordance with the present invention.

FIG. **2**A is a flowchart depicting one embodiment of a process for the ingress flow of data through a network switch.

FIG. **2**B is a flowchart depicting one embodiment of a process for the egress flow of data through a network switch.

FIG. **3**A is a flowchart depicting one embodiment of a process for mapping link channel data into virtual channel slots.

FIG. **3**B is a flowchart depicting one embodiment of a process for extracting payload packets.

FIG. **3**C is a flowchart depicting one embodiment of a process for mapping packet data into virtual channel slots.

FIG. **3**D is a flowchart depicting one embodiment of a process for mapping virtual channel slot data into link channels.

FIG. **4** is a flowchart depicting one embodiment of a process for a TSI switch to map slot data into outgoing slots.

FIG. **5** is a flowchart depicting one embodiment of a process for a TSI switch to forward slots.

FIG. **6** is a flowchart depicting an alternate embodiment of a process for the ingress flow of data through a network switch.

FIG. **7** is a flowchart depicting an alternate embodiment of a process for the egress flow of data through a network switch.

FIG. **8** is a block diagram depicting one embodiment of a mid-plane multiple card architecture for a network switch.

FIG. **9** is a block diagram depicting one embodiment of a control module.

FIG. **10** is a block diagram depicting one embodiment of a link interface.

FIG. **11** is a block diagram depicting an alternate embodiment of a link interface.

FIG. **12** is a block diagram depicting one embodiment of a processing engine.

FIG. **13** is a block diagram depicting one embodiment of a combined fabric and switch card.

FIG. **14** is a block diagram depicting one embodiment of a TSI switch.

DETAILED DESCRIPTION

FIG. **1** is a block diagram depicting one embodiment of network switch **90** in accordance with the present invention. Network switch **90** implements a mid-plane architecture to switch packets between multiple signaling mediums. Switch **90** supports multiple physical layer protocols in Layer 1 of the OSI model. Switch **90** also supports one or more higher-level protocols corresponding to Layer 2, Layer 3, and above in the OSI model. Switch **90** provides networking services that control the flow of data through switch **90**.

Switch **90** can be any type of network switch in various embodiments. In one embodiment, switch **90** is a network edge switch that provides Frame Relay, Gigabit Ethernet, Asynchronous Transfer Mode ("ATM"), and Internet Protocol ("IP") based services. In one example, switch **90** operates as a Provider Edge ("PE") Router implementing a virtual private network—facilitating the transfer of information between Customer Edge Routers that reside inside a customer's premises and operate as part of the same virtual private network. In another embodiment, switch **90** is a network core switch that serves more as a data conduit.

FIG. **1** shows that switch **90** has a mid-plane architecture that includes link interfaces **100**, **102**, **104**, and **106**, processing engines **110**, **112**, **114**, and **116**, switch **108**, and fabric **120**. In different embodiments, switch **90** can include more or less link interfaces and processing engines. In one embodiment, switch **90** includes 24 link interfaces and 12 processing engines. The link interfaces and processing engines are coupled to switch **108**, which switches data between the link interfaces and processing engines. The processing engines are also coupled to fabric **120**, which switches data between processing engines. In one implementation, fabric **120** is a switched fabric that switches packets between processing engines. In an alternative implementation, fabric **120** is replaced with a mesh of mid-plane traces with corresponding interfaces on the processing engines.

During ingress, data flows through an ingress link interface to switch **108**, which switches the data to an ingress processing engine. In one implementation, switch **108** is a multiplexing switch, such as a time slot based switch or packet based switch. The ingress processing engine processes the data and forwards it to an egress processing engine through fabric switch **120**. In one implementation, the ingress processing engine employs Layer 2 and Layer 3 lookups to perform the forwarding. The egress processing engine performs egress processing and forwards data to switch **108**. Switch **108** switches the data to an egress link interface for transmission onto a medium. More details regarding the ingress and egress flow of data through switch **90** are provided below with reference to FIGS. **2**A, **2**B, **6**, and **7**.

Each link interface exchanges data with one or more physical networking mediums. Each link interface exchanges data with the mediums according to the Layer 1 physical signaling standards supported on the mediums. In some embodiments, a link interface also performs a portion of Layer 2 processing, such as MAC framing for Gigabit Ethernet.

In one example, link interfaces **100** and **106** interface with mediums **122** and **128**, respectively, which carry STS-48 SONET over OC-48; link interface **102** interfaces with medium **124**, which carries channelized SONET over OC-48, such as 4 STS-12 channels; and link interface **104** interfaces with medium **126**, which carries Gigabit Ethernet. In various embodiments, many different physical mediums, physical layer signaling standards, and framing protocols can be supported by the link interfaces in switch **90**.

The processing engines in switch **90** deliver the services provided by switch **90**. In one implementation, each processing engine supports multiple Layer 2, Layer 3, and higher-level protocols. Each processing engine in switch **90** processes packets or cells in any manner supported by switch **90**—allowing any processing engine to service data from any medium coupled to a link interface in switch **90**. In one embodiment, processing engine operations include Layer 2 and Layer 3 switching, traffic management, traffic policing, statistics collection, and operation and maintenance ("OAM") functions.

In one implementation, switch **108** is a TSI switch that switches data streams between the link interfaces and processing engines. In one embodiment, TSI switch **108** switches time slots of data between link interfaces and processing engines. In one implementation, each time slot can support a single STS-1 channel. One version of TSI switch **108** interfaces with link interfaces and processing engines through TSI switch ports. During ingress, an ingress link interface maps incoming data into a set of time slots and passes the set of time slots to an incoming TSI switch port in TSI switch **108**. In one implementation, TSI switch **108** supports an incoming set of time slots with 48 unique slots, each capable of carrying bandwidth for an STS-1 channel of a SONET frame. In one such embodiment, TSI switch **108** receives the incoming set of time slots on an incoming TSI switch port. In different embodiments, different time slot characteristics can be employed. TSI switch **108** switches the received time slots into outgoing time slots for delivery to ingress processing engines. TSI switch **108** delivers outgoing time slots for each ingress processing engine through an outgoing TSI switch port associated with the respective ingress processing engine.

During egress, an egress processing engine maps egress data into time slots and passes the time slots to TSI switch **108**, which receives the time slots into an incoming TSI switch port. TSI switch **108** maps the received time slots into outgoing time slots for delivery to egress link interfaces through outgoing TSI switch ports. In one implementation, TSI switch **108** includes the following: (1) an incoming TSI switch port for each ingress link interface and each egress processing engine, and (2) an outgoing TSI switch port for each ingress processing engine and each egress link interface. When a link interface or processing engine performs both ingress and egress operations, TSI switch **108** includes an incoming TSI switch port and an outgoing TSI switch port for the link interface or processing engine.

In one implementation, TSI switch **108** performs time division multiplexing. TSI switch **108** is capable of switching a time slot in any incoming set of time slots to any time slot in any outgoing set of time slots. Any time slot from a link interface can be delivered to any processing engine. Any time slot from a processing engine can be delivered to any link interface that supports the protocol for the time slot's data. This provides a great deal of flexibility when switching data between link interfaces and processing engines—allowing data to be switched so that no processing engine or link interface becomes over utilized while others remain under utilized.

In an alternate embodiment, switch **108** is a packet switch. In this embodiment, the link interfaces and processing engines deliver data to packet switch **108** in the form of packets with headers. Packet switch **108** uses the headers to switch the packets to the appropriate link interface or processing engine.

FIG. **2A** is a flowchart depicting one embodiment of a process for the ingress flow of data through network switch **90** when switch **108** is a TSI switch. An ingress link interface, such as link interface **100**, **102**, **104**, or **106**, receives physical signals over a medium, such as link **122** (step **10**). The physical signals conform to a physical signaling standard from Layer 1 of the OSI model. In one implementation, each link interface includes one or more transceivers to receive the physical signals on the medium in accordance with the Layer 1 protocol governing the physical signaling. Different link interfaces in network switch **90** can support different Layer 1 physical signaling standards. For example, some link interfaces may support OC-48 physical signaling, while other link interfaces support physical signaling for Gigabit Ethernet. The reception process (step **10**) includes the Layer 1 processing necessary to receive the data on the link.

In one implementation, each link supported by a link interface includes one or more channels. In one example, link **122** is an OC-48 link carrying **4** separate STS-12 channels. In different embodiments, a link can have various channel configurations. A link can also carry only a single channel of data. For example, link **122** can be an OC-48 link with a single STS-48 channel.

The ingress link interface maps data from incoming link channels into virtual channel time slots in switch **90** (step **12**). As described above, one embodiment of switch **90** employs TSI switch **108** to pass data from ingress link interfaces to ingress processing engines. Each ingress link interface maps

data from link channels to time slots that are presented to TSI switch **108** for switching. In one embodiment, each link interface maps link channel data into a set of 48 time slots for delivery to TSI switch **108**.

Switch **90** employs virtual concatenation of time slots to form virtual channels within switch **90**. Each time slot is assigned to a virtual channel. In some instances, multiple time slots are assigned to the same virtual channel to create a single virtual channel with increased bandwidth. In one example, each time slot has the ability to support bandwidth for a STS-1 channel of data. When a single time slot is assigned to a virtual channel, the virtual channel is an STS-1 channel. When multiple time slots are assigned to the same virtual channel, the resulting virtual channel operates as a single channel with the bandwidth of a single STS-X channel—X is the number of time slots assigned to the virtual channel. When multiple time slots are assigned to a single virtual channel there is no requirement for the assigned time slots to be adjacent to one another. However, the time slots can be adjacent in some embodiments. In further embodiments, time slots can support a channel bandwidth other than a STS-1 channel.

In various embodiments, different techniques can be employed for mapping link channel data into virtual channel time slots. In one example, a link includes 4 STS-12 channels, and the ingress link interface coupled to the link supports 4 STS-12 virtual channels—4 virtual channels each being assigned 12 time slots with STS-1 bandwidth. In this example, the ingress link interface maps data from each STS-12 link channel to a respective one of the 4 STS-12 virtual channels. FIG. **3A** shows a process that employs Layer 2 framing before mapping link channel data into virtual channel time slots. More details regarding FIG. **3A** are provided below.

The ingress link interface forwards the set of time slots to TSI switch **108** (step **14**). In one implementation, each ingress link interface supports a set of 48 time slots, and TSI switch **108** receives a set of 48 time slots from each ingress link interface. In this embodiment, each ingress link interface forwards the set of 48 time slots to TSI switch **108** in the form of GFP framed data over SONET. In alternate embodiments, switch **90** employs different numbers of time slots and different methods of forwarding time slots to TSI switch **108**.

TSI switch **108** switches the incoming time slots from ingress link interfaces to outgoing time slots for delivery to ingress processing engines (step **16**). TSI switch **108** forwards sets of outgoing time slots to their respective ingress processing engines (step **18**). There is an outgoing set of time slots associated with each ingress processing engine coupled to TSI switch **108**. TSI switch **108** maps each incoming time slot from an ingress link interface to a time slot in an outgoing set of time slots for an ingress processing engine. TSI switch **108** has the ability to direct any incoming time slot of data from a link interface to any processing engine on any time slot in any outgoing set of time slots.

TSI switch **108** can map time slot data from an incoming set of time slots to time slots in multiple outgoing sets of time slots—a first time slot in an incoming set of time slots can be mapped to a time slot in one outgoing set of time slots and a second time slot in the incoming set of time slots can be mapped to a time slot in a different outgoing set of time slots. TSI switch **108** can also map time slots from different incoming sets of time slots to time slots in the same outgoing set of time slots—a time slot in a first incoming set of time slots can be mapped to a time slot in an outgoing set of time slots and a time slot in a different incoming set of time slots can be mapped to a time slot in the same outgoing set of time slots.

In one implementation, each ingress processing engine is assigned 48 outgoing time slots. TSI switch **108** maps the data from each incoming time slot to one of the 48 time slots for one of the ingress processing engines. TSI switch **108** forwards each outgoing set of 48 time slots to a respective ingress processing engine in the form of GFP framed data over SONET. In alternate implementations, an outgoing set of time slots can have a different format than the incoming set of time slots. More details regarding the mapping performed by TSI switch **108** appears below.

An ingress processing engine, such as processing engines **110**, **112**, **114**, or **116**, receives an outgoing set of time slots from TSI switch **108** and extracts payload data packets (step **20**). The payload data is the data carried within each virtual channel. An ingress processing engine extracts the payload data and maps the data into packets that can be processed according to the protocols supported on the processing engine. In one implementation, one or more processing engines each support multiple protocols within each layer of the OSI model. In another implementation, one or more processing engines each support all protocols supported by the processing engines in switch **90** within each layer of the OSI model supported on the processing engines. These implementations allow an ingress processing engine to perform different processing on data from each of the virtual channels received via the outgoing set of time slots from TSI switch **108**. In yet another embodiment, a processing engine does not support multiple protocols within each layer of the OSI model. Further details regarding the extraction of payload data are provided below with reference to FIG. **3B**.

The ingress processing engine processes the extracted payload data packets according to the identified protocol for the data (step **22**). Payload data received from one time slot may require different processing than payload data received from a different time slot. In one embodiment, the ingress processing engine performs data processing at Layer 2 and Layer 3 of the OSI model. In further embodiments, the ingress processing engine may perform processing at Layer 2, Layer 3 and above in the OSI model.

The ingress processing engine generates fabric cells for delivering processed data to an egress processing engine through fabric **120** (step **24**). In one implementation, the ingress processing engine generates fabric cells by breaking the payload data associated with processing packets into smaller cells that can be forwarded to fabric **120**. Various fabric cell formats can be employed in different embodiments. The ingress processing engine formats the cells according to a standard employed for delivering cells to fabric **120**. Those skilled in the art will recognize that many different well-known techniques exist for formatting fabric cells. The ingress processing engine forwards the fabric cells to fabric **120** (step **26**).

FIG. **2B** is flowchart depicting one embodiment of a process for the egress flow of data through network switch **90** when switch **108** is a TSI switch. Fabric **120** forwards fabric cells to an egress processing engine, such as processing engine **110** and **112**, **114**, or **116** (step **30**). The egress processing engine reassembles the fabric cells into one or more processing packets of data (step **32**). The egress processing engine processes the packets according to the appropriate OSI model protocols. In one implementation, the egress processing engine performs Layer 2 and Layer 3 processing. In alternate implementations, there is no need for packet processing on the egress processing engine.

The egress processing engine maps processing packet data into virtual channel slots (step **36**) and forwards the virtual channel slots to TSI switch **108** (step **38**). On each egress processing engine, each virtual channel is represented by one or more time slots in a set of time slots. In one embodiment, each time slot can support the bandwidth of a STS-1 channel. In one implementation, the set of time slots includes 48 time slots, and the egress processing engine forward the 48 time slots to TSI switch **108** in the form of GFP framed data over SONET. In alternate embodiments, different time slot sizes can be employed and different mechanisms can be employed for forwarding sets of time slots. More details regarding the mapping of packet data into virtual channel slots is provided below with reference to FIG. 3C.

TSI switch **108** switches the incoming set of time slots from each egress processing engine (step **40**). TSI switch **108** maps each time slot in an incoming set of time slots into a time slot in an outgoing set of time slots for delivery to an egress link interface. This mapping process is the same as described above for mapping data from ingress link interface time slots into outgoing sets of time slots for ingress processing engines (step **16**, FIG. **2A**). TSI switch **108** is capable of mapping any time slot from an egress processing engine set of time slots to any time slot of any outgoing set of time slots for any egress link interface. TSI switch **108** forwards outgoing sets of time slots to the appropriate egress link interfaces (step **42**). In one implementation, an outgoing set of slots is in the form of GFP framed data over SONET. Different forwarding formats and time slot sizes can be employed in various embodiments.

An egress link interface that receives an outgoing set of time slots from TSI switch **108** maps virtual channel slot data into link channels (step **44**). FIG. **3D** shows a flowchart for one method of carrying out step **44** by framing virtual channel data and mapping the framed data into link channels. In alternate embodiments, different techniques can be employed to carryout step **44**. The egress link interface transmits the physical signals for data in each link channel as physical signals on the medium coupled to the link interface (step **46**). The link interface transmits the frames according to the Layer 1 signaling protocol supported on the medium.

FIG. **3A** is a flowchart describing one embodiment of a process for mapping link channel data into virtual channel slots (step **12**, FIG. **2A**). The ingress link interface maps link channel data into frames (step **50**). In one implementation, the ingress link interface performs Layer 2 processing on incoming link data to create Layer 2 frames. In alternate embodiments, different protocol rules can be employed to generate frames from link channel data. In one embodiment, switch **90** maintains mapping tables that are used by ingress processing engines to map link channel data into frames.

One implementation of the table contains entries with the following fields: 1) Link Channel—identifying a link channel for the ingress processing engine; 2) Protocol—identifying a Layer 1 and Layer 2 protocol for the identified link channel; and 3) Frame—identifying one or more frames to receive data from the identified link channel. When data arrives at an ingress link interface, the link interface uses the table entry that corresponds to the link channel supplying the data. The ingress link interface maps the data into the identified frames using the identified Layer 1 and Layer 2 protocols. Each link channel can be programmed for a different Layer 1 and/or Layer 2 protocol. A user of switch **90** programs the fields in the above-identified table in one embodiment. In further embodiments, different fields can be employed and mechanisms other than a mapping table can be employed.

The ingress processing engine maps the frame data into virtual channels (step **51**) and maps the virtual channel's data into time slots in a set of time slots the link interface will forward to TSI switch **108** (step **52**). In one implementation, these steps are performed as separate operations. In alternate implementations, these steps are combined into a single step.

In one embodiment, network switch **90** maintains mapping tables that are used by the ingress link interface to map incoming data into virtual channels and virtual channel data into a set of time slots. In one implementation, the table contains entries with the following fields: 1) Virtual Channel—identifying a virtual channel; 2) Time Slots—identifying all time slots in the ingress link interface's set of time slots that belong to the identified virtual channel; 3) Link Channel—identifying one or more link channels that are to have their data mapped into the identified virtual channel; and 4) Link Channel Protocol—identifying the Layer 1 and Layer 2 protocols employed for the data in the identified link channels. In one implementation, the Link Channel field can identify one or more frames formed as a result of step **50**. Alternatively, different information can be used to identify link channel data for a virtual channel when the ingress link interface does not frame link channel data. A user of switch **90** programs these fields in one embodiment. In further embodiments, different fields can be employed and mechanisms other than a mapping table can be employed.

The ingress link interface uses a table entry to map data into a virtual channel. The ingress link interface maps data from the entry's identified link channel into the entry's identified time slots for the virtual channel. The ingress link interface formats the link channel data in the virtual channel time slots, based on the entry's identified Layer 1 and Layer 2 protocols for the link channel data.

FIG. **3B** is a flowchart describing one embodiment of a process for extracting payload packets (step **20**, FIG. **2A**). The egress processing engine maps data from each time slot received from TSI switch **108** into a virtual channel (step **53**) and maps data from the virtual channel into one or more payload data packets for processing (step **54**). In one implementation, these steps are performed as separate operations. In alternate implementations, these steps are combined into a single step.

In one embodiment, network switch **90** maintains mapping tables that are used by the ingress processing engine to map incoming slot data to packets for processing by the ingress processing engine. In one implementation, the table contains an entry for each virtual channel. Each entry includes the following fields: 1) Virtual Channel—identifying a virtual channel; 2) Time Slots—identifying all time slots in the ingress processing engine's set of time slots that belong to the identified virtual channel; 3) Link Channel—identifying one or more link channels that are to have their data mapped into the identified virtual channel; and 4) Link Channel Protocol—identifying the Layer 1 and Layer 2 protocols employed for the data in the identified link channels. A user of switch **90** programs these fields in one embodiment. In further embodiments, different fields can be employed and mechanisms other than a mapping table can be employed.

The ingress processing engine uses this table to extract payload data into packets for processing. When each time slot of data arrives at the ingress processing engine, the ingress processing engine associates the time slot with a virtual channel in an entry corresponding to the time slot. The ingress processing engine parses the contents of the virtual channel to obtain payload data for processing packets. The ingress processing engine uses the information in the Link Channel Protocol field to parse the virtual channel. The ingress processing engine also places information in a header of each processing packet that identifies the link channel associated with the virtual channel being mapped into the packet. This

information will be useful when directing the processing packet's contents through the egress flow described above.

FIG. **3C** is a flowchart depicting one embodiment of a process for mapping packet data into virtual channel slots (step **36**, FIG. **2B**). The egress processing engine maps processing packet data into virtual channels (step **55**) and maps virtual channel data into time slots (step **56**) for delivery to TSI switch **108**. In one implementation, these steps are performed as separate operations. In alternate implementations, these steps are combined into a single step.

In one embodiment, network switch **90** maintains mapping tables that are used by the egress processing engine to map packet data into virtual channel time slots. In one implementation, the table contains an entry for each virtual channel. Each entry includes the following fields: 1) Virtual Channel—identifying a virtual channel; 2) Time Slots—identifying all time slots in the egress processing engine's set of time slots that belong to the identified virtual channel; 3) Link Channel—identifying one or more link channels that are to have their data mapped into the identified virtual channel; and 4) Link Channel Protocol—identifying the Layer 1 and Layer 2 protocols employed for the data in the identified link channels. A user of switch **90** programs these fields in one embodiment. In further embodiments, different fields can be employed and mechanisms other than a mapping table can be employed.

The egress processing engine identifies the link channel that is intended to receive a processing packet's data. In one implementation, the processing packet's header includes this information. The egress processing engine identifies the table entry that corresponds to the link channel. The egress processing engine uses the entry to identify the corresponding virtual channel and associated time slots. The egress processing engine maps the packet data into these virtual channel time slots, based on the protocols identified in the Link Channel Protocol field.

FIG. **3D** is a flowchart depicting one embodiment of a process for mapping virtual channel slot data into link channels (FIG. **44**, FIG. **2B**). The egress link interface maps time slot data from TSI switch **108** into virtual channels (step **57**) and maps virtual channel data into frames (step **58**). In one implementation, these steps are performed as separate operations. In alternate implementations, these steps are combined into a single step.

In one embodiment, network switch **90** maintains mapping tables that are used by the ingress link interface to map slot data into virtual channels and virtual channel data into frames. In one implementation, the table contains an entry for each virtual channel. Each entry includes the following fields: 1) Virtual Channel —identifying a virtual channel; 2) Time Slots—identifying all time slots in the egress link interface's set of time slots that belong to the identified virtual channel; 3) Link Channel—identifying one or more link channels that are to have their data mapped into the identified virtual channel; and 4) Link Channel Protocol—identifying the Layer 1 and Layer 2 protocols employed for the data in the identified link channels. A user of switch **90** programs these fields in one embodiment. In further embodiments, different fields can be employed and mechanisms other than a mapping table can be employed.

The egress link interface uses this table to map time slot data into virtual channels. For a time slot that arrives from TSI switch **108**, the egress link interface maps data into the identified virtual channel for the time slot. For each virtual channel, the egress link interface maps the channel's data into the link interface identified for the virtual channel. For the framed data embodiment described above, the egress link interface maps the virtual channel data into one or more frames that correspond to the identified link channel. These frames can be identified as part of the Link Channel field in one embodiment. The egress link interface formats the virtual channel data in the frames, based on the identified Layer 1 and Layer 2 protocols for the link channel data.

In the frame data implementation, the egress link interface maps frame data into link channels (step **59**). In one embodiment, switch **90** maintains mapping tables used by egress processing engines to map frame data into link channels. One implementation of the table contains entries for each link channel, including the following fields: 1) Link Channel—identifying a link channel for the egress processing engine; 2) Protocol—identifying Layer 1 and Layer 2 protocols for the identified link channel; and 3) Frame—identifying one or more frames that maintain data from the identified link channel. When virtual channel data is framed, the egress link interface uses the table entry that corresponds to a selected frame. The egress link interface maps the frame data into the identified channel using the identified Layer 1 and Layer 2 protocols. A user of switch **90** programs the fields in the above-identified table in one embodiment. In further embodiments, different fields can be employed and mechanisms other than a mapping table can be employed.

FIG. **4** is a flowchart depicting one embodiment of a process for TSI switch **108** to map slot data into outgoing slots. Ingress link interfaces and egress processing engines forward sets of time slots to TSI switch **108**. In one implementation, slots are sent to TSI switch **108** in the form of GFP framed data over SONET. In one embodiment, TSI switch **108** receives a set of 48 time slots from each ingress link interface and egress processing engine.

TSI switch **108** receives an incoming time slot (step **60**). TSI switch **108** determines whether the slot has idle data (step **61**). If the slot is idle, TSI switch **108** loops back to step **60** to receive the next slot. If the slot is not idle, TSI switch **108** maps the data in the slot to a slot in an outgoing set of slots (step **62**). TSI switch **108** maps data from an ingress link interface to a slot in an outgoing set of slots for an ingress processing engine. TSI switch **108** maps data from an egress processing engine to a slot in an outgoing set of slots for an egress link interface. TSI switch **108** returns to step **60** to receive the next incoming slot.

In one embodiment, TSI switch **108** employs a mapping table to map incoming slot data to a slot in an outgoing set of slots (step **62**). One example of a mapping table includes entries with the following fields: 1) Incoming Port—identifying an incoming TSI switch port on TSI switch **108** that is coupled to either an ingress link interface or egress processing engine to receive a set of time slots; 2) Incoming Slot—identifying a time slot in the incoming set of time of slots on the identified incoming TSI switch port; 3) Outgoing Port—identifying an outgoing TSI switch port on TSI switch **108** that is coupled to either an ingress processing engine or egress link interface to provide an outgoing set of slots; and 4) Outgoing Slot—identifying a time slot in the outgoing set of slots for the identified outgoing TSI switch port.

When time slot data is received, TSI switch **108** finds a corresponding table entry. The corresponding table entry has an Incoming Port field and Incoming Slot field that correspond to the port on which the incoming set of slots is being received and the slot in the incoming set of slots that is being received. TSI switch **108** maps the incoming slot data to a slot in an outgoing set of slots that is identified by the entry's Outgoing Port and Outgoing Slot fields. In one implementation, each outgoing set of slots corresponds to an outgoing TSI switch port in TSI switch **108**. The outgoing TSI switch port is coupled to deliver the outgoing set of slots to either an egress link interface or ingress processing engine.

In alternate embodiments, different mapping table formats can be employed. For example, each incoming TSI switch port in the TSI switch has its own mapping table in one embodiment—including the Incoming Slot, Outgoing Port, and Outgoing Slot fields. Alternatively, the Outgoing Port field can be modified to identify a transmit port that corresponds to a set of slots. In different embodiments, the mapping table is replaced by a different instrumentality that serves the same purpose.

In a further implementation, the above-described mapping table includes the following additional fields: 5) Backup Outgoing Port—identifying a backup outgoing TSI switch port for the port identified in the Outgoing Port field; 6) Backup Outgoing Slot—identifying a slot in the outgoing set of slots for the port identified in the Backup Outgoing Port field; and 7) Backup—indicating whether to use the Outgoing Port and Outgoing Slot fields or the Backup Outgoing Port and Backup Outgoing Slot fields. A user of switch **90** sets values in these fields in one implementation. In an alternate embodiment, these backup fields are maintained in a central memory of switch **90** and backup values are loaded into the above-described table only when a backup is needed.

These additional table fields can be used to support redundancy. A link interface or processing engine associated with an outgoing set of slots may become disabled. When this happens, TSI switch **108** will use the Backup Outgoing Port and Backup Outgoing Slot fields in place of the Outgoing Port and Outgoing Slot fields. This provides great flexibility in creating redundancy schemes on a per channel basis, per time slot basis, per port basis, per group of ports basis, or other basis. If a link interface fails, the virtual channel slots associated with the failed link interface can be redistributed among multiple link interfaces. Similarly, if a processing engine fails, the virtual channel slots associated with the failed processing engine can be redistributed among multiple processing engines. Switch **90** implements the redistribution by modifying the mapping information in the mapping table—switch **90** sets values in the above-described Backup fields to control the mapping operation of switch **108**. This flexibility allows redundancy to be shared among multiple link interfaces and processing engines. In fact, network switch **90** can avoid the traditional need of having an entire link interface PCB and an entire processing engine PCB set aside for redundancy purposes. Switch **90** can modify mapping information automatically, upon detecting a condition that calls for modification. Alternatively, a user can manually alter mapping information.

Efficiency is greatly increased when each processing engine supports all protocols used in switch **90** at the OSI model layers supported by the processing engines. In this embodiment, each processing engine can receive and process data from any time slot in any link interface's set of time slots. This allows backup processing engines to be assigned so that no processing engine becomes over utilized and no processing engine remains under utilized. In one implementation, switch **90** modifies mapping information by setting values in the Backup fields to facilitate efficient bandwidth pooling. Switch **90** monitors the utilization of processing engines and link interfaces. If any link interface or processing engine becomes over or under utilized, switch **90** sets values in the above-described Backup fields to redirect the flow of data to make link interface and processing engine utilization more evenly distributed.

In a further embodiment, switch **90** employs the above-described Backup field to implement 1:N, 1:1, or 1+1 redundancy. In 1:N redundancy, a time slot or set of time slots is reserved for backing up a set of N time slots. In 1:1 redundancy, each time slot or set of time slots is uniquely backed up by another time slot or set of time slots. In 1+1 redundancy, an incoming time slot is mapped to two outgoing time slots one time slot identified by the Outgoing Port and Outgoing Slot fields, and another time slot identified by the Backup Outgoing Port and Backup Outgoing Slot fields. This allows redundant dual paths to be created through switch **90**. The ability of switch **90** to efficiently distribute processing engine resources allows this dual path redundancy to be achieved without significant decrease in the overall throughput performance of switch **90**.

FIG. **5** is a flowchart depicting one embodiment of a process for TSI switch **108** to forward slots in an outgoing set of slots. TSI switch **108** selects a slot (step **64**). TSI switch **108** determines whether the slot is to contain an idle signal or valid virtual channel slot data (step **65**). If the slot is to be idle, TSI switch **108** maps an idle data pattern into the selected slot (step **67**) and forwards the slot to an ingress processing engine or egress link interface (step **68**). If the slot is not idle (step **65**), TSI switch **108** maps virtual channel data into the selected slot (step **66**) and forwards the slot to an ingress processing engine or egress link interface (step **68**). TSI switch **108** continues to loop back to step **64** and repeat the above-described process.

In one implementation, the process in FIG. **5** can be performed in real time while the outgoing set of slots is being forwarded. Alternatively, an entire outgoing set of slots is assembled before forwarding any channels.

FIG. **6** is a flowchart depicting an alternate embodiment of a process for the ingress flow of data through network switch **90** when switch **108** is a packet switch. The process steps with the same numbers as those appearing in FIG. **2A** operate the same as described for FIG. **2A**. The description in FIG. **6** will highlight the differences in the ingress data flow when packet switch **108** is employed.

After the ingress link interface receives physical signals for link channels (step **10**), the ingress processing engine maps link channel data into one or more packets (step **70**). The link interface forwards each packet to packet switch **108** (step **72**) for delivery to an ingress processing engine. Each packet includes a payload and a header. The payload includes the data received from the physical medium that needs to be forwarded to an ingress processing engine. The header includes information necessary for packet switch **108** to properly direct the packet to a targeted ingress processing engine. The ingress link interface creates the header in the step of mapping data into the packet (step **70**).

In one implementation, the header includes the following fields: 1) Destination PE—identifying the targeted ingress processing engine; 2) Source LI—identifying the ingress link interface that created the packet; 3) Source PHY—identifying the link interface transceiver that received the data in the packet's payload; and (4) Source Channel—identifying a link channel in which the payload data was received by the ingress link interface. In alternate embodiments, different header fields can be employed.

In one implementation, the ingress link interface maps data into the packet's payload (step **70**) using a mapping table. One embodiment of the mapping table includes entries with the following fields: 1) Destination—identifying a processing engine; 2) Link Channel—identifying a link channel; and 3) Protocol—identifying the Layer 1 and Layer 2 protocols format of data in the identified link channel. A user of switch **90** programs these fields in one embodiment. The ingress link interface maps data into a packet from a link channel. The

ingress link interface identifies a table entry that corresponds to the link channel and uses the protocols specified in the entry's Protocol field to move data from the link channel to the packet. The ingress link interface also loads the Destination PE field in the packet header with the processing engine identified in the entry's Destination field.

Packet switch **108** identifies the targeted ingress processing engine for the packet (step **74**) and forwards the packet to the targeted ingress processing engine (step **76**). In one implementation, packet switch **108** uses the Destination PE field in the packet header to identify the targeted ingress processing engine. The ingress processing engine extracts payload data in the packets from packet switch **108** (step **77**). The ingress processing engine maps the payload data into processing packets for processing by the ingress processing engine.

In one embodiment, network switch **90** maintains mapping tables that are used by the ingress processing engine to map payload data from the ingress processing engine into processing packets (step **77**). In one implementation, the table contains entries with the following fields: 1) Source Information—identifying a permutation of values from the packet header fields Source LI, Source PHY, and Source Channel; 2) Protocol—identifying the Layer 1 and Layer 2 protocols associated with the data having a header that matches the Source Information field; and 3) Link Channel—identifying the link channel that originated the data. A user of switch **90** programs these fields in one embodiment. In alternate embodiments, different fields can be employed, or other instrumentalities can replace the table.

When a packet arrives from packet switch **108**, the ingress processing engine finds an entry with a Source Information field that corresponds to the values in the packet's header. The ingress processing engine then uses the identified entry's Protocol field to map the packet payload data into a processing packet. In one implementation, the ingress processing engine also includes a link channel identifier in the processing packet, based on the Link Channel field. The remaining steps in FIG. **6** conform to those described above for FIG. **2**.

FIG. **7** is a flowchart depicting an alternate embodiment of a process for the egress flow of data through network switch **90** when switch **108** is a packet switch. The steps in FIG. **7** with the same reference numbers as those in FIG. **3** operate in the same manner described for FIG. **3**. The description of FIG. **7** will highlight the differences in the egress data flow when switch **108** is a packet switch.

After processing packets from reassembled fabric cells (Step **34**), the egress processing engine maps packet data into new packets for delivery to packet switch **108** (step **80**). In one implementation, the egress processing engine uses a mapping table to perform this operation. One embodiment of the mapping table includes the following fields: 1) Packet Information—identifying information to use in a packet header; 2) Link Channel—identifying a link channel that originated the data being put into the packet; and 3) Protocol—identifying the Layer 1 and Layer 2 protocols for the packet data. A user of network switch **90** configures these fields. In alternate embodiments, different fields can be employed, or the table can be replaced by a different instrumentality.

The egress processing engine identifies a table entry that has a Link Channel field that corresponds to the link channel that originated the payload data in the processing packet. The egress processing engine maps the payload data into a packet for packet switch **108**, based on the protocols in the corresponding Protocol field. The egress processing engine uses the entry's Packet Information field to create a header for the packet. In one implementation, the packet headers include the following fields: 1) Source PE—identifying the egress processing engine that created the packet; 2) Destination LI—identifying a targeted egress link interface for the packet 3) Destination PHY—identifying a targeted transceiver on the identified egress link interface; and (4) Destination Channel—identifying a targeted link channel in which the payload data is to be transmitted from the egress link interface. In alternate embodiments, different header fields can be employed.

The egress processing engine forwards the new packets to packet switch **108** for switching to the targeted egress link interface (step **82**). Packet switch **108** identifies the targeted egress link interface for the incoming packet (step **84**). Packet switch **108** uses the header information in the packet to make this identification. For the header described above, the Destination LI field identifies the targeted egress link interface. Packet switch **108** forwards the packet to the targeted egress link interface (step **86**). Transmission data frames are generated (step **87**) and physically transmitted (step **46**). In order to generate frames (step **87**), the egress processing engine uses the header fields in the packet from packet switch **108**. In one implementation, packet switch **108** uses the Destination PHY and Destination Channel fields to generate these frames.

FIG. **8** is a block diagram depicting one embodiment of switch **90**, implemented with a mid-plane architecture. Switch **90** includes control module **130**, which is coupled to control bus **150**. The above-described link interfaces **100**, **102**, **104**, and **106**, processing engines, **110**, **112**, **114**, and **116**, switch **108**, and fabric **120** are also coupled to control bus **150**. Control bus **150** carries control information for directing the operation of components in switch **90**. In one implementation, control bus **150** is a 100 Base-T Ethernet communication link. In such an embodiment, control bus **150** employs the 100 Base-T Ethernet protocols for carrying and formatting data. In alternate embodiments, a variety of different protocols can be employed for implementing control bus **150**. In a further embodiment, control bus **150** is a star-like switched Ethernet network. Further details regarding the operation of control module **130** are provided below.

Link interfaces **100**, **102**, **104**, and **106**, processing engines **110**, **112**, **114**, and **116**, and switch **108** are coupled to switch plane **152**. Switch plane **152** carries sets of time slots. In one implementation, switch **108** is a TSI switch and switch plane **152** carries GFP framed data over SONET. In one such implementation, the capacity of switch plane **152** is 2.488 Giga-bits per second, with the SONET frame containing 48 time slots that each support bandwidth equivalent to one STS-1 channel. Alternatively, the frame may include higher bandwidth channels or even different size channels. In further embodiments, switch plane **152** carries STS-192 SONET. In another embodiment, switch **108** is a packet switch and switch plane **152** carries packets. Processing engines **110**, **112**, **114**, and **116** and fabric **120** are coupled to fabric plane **154**. The processing engines and fabric **120** exchange fabric cells across fabric plane **154**.

FIG. **8** shows data planes **152** and **154** as separate from control bus **150**. In alternate embodiments, control bus **150** can be implemented as part of switch plane **152** and fabric plane **154**.

FIG. **9** is a high-level block diagram depicting one embodiment of control module **130**. In one embodiment, control module **130** is a PCB in switch **90**. Control module **130** directs the operation of link interfaces **100**, **102**, **104**, and **106**, processing engines **110**, **112**, **114**, and **116**, switch **108**, and fabric **120**. In one implementation, control module **130** directs the operation of these components by issuing configuration and operation instructions that dictate how the components operate.

Control module **130** also maintains a management information base ("MIB") that maintains the status of each component at various levels of detail. In one implementation, the MIB maintains information for each link interface and each processing engine. This enables control module **130** to determine when a particular component in switch **90** is failing, being over utilized, or being under utilized. Control module **130** can react to these determinations by making adjustments in the internal switching of data between link interfaces and processing engines through switch **108**—changing switching of data associated with failed or inefficiently utilized components.

In one example, control module **130** detects that a processing engine is under utilized. Control module **130** responds by arranging for switch **108** to switch one or more time slots from one or more link interfaces to the under utilized processing engine. In another example, control module **130** determines that a failure occurred at a link interface. Control module **130** arranges for switch **108** to switch each time slot originally directed to the failed link interface to one or more different link interface. In one implementation, the time slots are distributed to several alternative link interfaces. Control module **130** facilitates the above-described time slot switching changes by modifying mapping table information, such as the Backup field, in one embodiment. The mapping table information can be maintained in control module **130** or distributed on switch **108**.

In one embodiment, control module **130** contains processing unit **205**, main memory **210**, and interconnect bus **225**. Processing unit **205** may contain a single microprocessor or a plurality of microprocessors for configuring control module **130** as a multi-processor system. Processing unit **205** is employed in conjunction with a memory or other data storage medium containing application specific program code instructions to implement processes carried out by switch **90**.

Main memory **210** stores, in part, instructions and data for execution by processing unit **205**. If a process is wholly or partially implemented in software, main memory **210** can store the executable instructions for implementing the process. In one implementation, main memory **210** includes banks of dynamic random access memory (DRAM), as well as high-speed cache memory.

Control module **130** further includes control bus interface **215**, mass storage device **220**, peripheral device(s) **230**, portable storage medium drive(s) **240**, input control device interface **270**, graphics subsystem **250**, and output display interface **260**, or a subset thereof in various embodiments. For purposes of simplicity, all components in control module **130** are shown in FIG. **9** as being connected via bus **225**. Control module **130**, however, may be connected through one or more data transport means in alternate implementations. For example, processing unit **205** and main memory **210** may be connected via a local microprocessor bus. Control bus interface **215**, mass storage device **220**, peripheral device(s) **230**, portable storage medium drive(s) **240**, and graphics subsystem **250** may be coupled to processing unit **205** and main memory **210** via one or more input/output busses.

Mass storage device **220** is a non-volatile storage device for storing data and instructions for use by processing unit **205**. Mass storage device **220** can be implemented in a variety of ways, including a magnetic disk drive or an optical disk drive. In software embodiments of the present invention, mass storage device **220** stores the instructions executed by control module **130** to perform processes in switch **90**.

Portable storage medium drive **240** operates in conjunction with a portable non-volatile storage medium to input and output data and code to and from control module **130**.

Examples of such storage mediums include floppy disks, compact disc read only memories (CD-ROM) and integrated circuit non-volatile memory adapters (i.e. PC-MCIA adapter). In one embodiment, the instructions for control module **130** to execute processes in switch **90** are stored on such a portable medium, and are input to control module **130** via portable storage medium drive **240**.

Peripheral device(s) **230** may include any type of computer support device, such as an input/output interface, to add additional functionality to control module **130**. For example, peripheral device(s) **230** may include a communications controller, such as a network interface, for interfacing control module **130** to a communications network. Instructions for enabling control module **130** to perform processes in switch **90** may be downloaded into main memory **210** over a communications network. Control module **130** may also interface to a database management system over a communications network or other medium that is supported by peripheral device(s) **230**.

Input control device interface **270** provides interfaces for a portion of the user interface for control module **130**. Input control device interface **270** may include an alphanumeric keypad for inputting alphanumeric and other key information, a cursor control device, such as a mouse, a trackball, stylus, or cursor direction keys. In order to display textual and graphical information, control module **130** contains graphics subsystem **250** and output display interface **260**. Output display interface **260** can include an interface to a cathode ray tube display or liquid crystal display. Graphics subsystem **250** receives textual and graphical information, and processes the information for output to output display interface **260**.

Control bus interface **215** is coupled to bus **225** and control bus **150**. Control bus interface **215** provides signal conversion and framing to support the exchange of data between bus **225** and control bus **150**. In one implementation, control bus interface **215** implements **100** Base-T Ethernet protocols—converting data between the format requirements of bus **225** and the 100 Base-T Ethernet format on control bus **150**.

FIG. **10** is a block diagram of one embodiment of a link interface in switch **90**, such as link interface **100**, **102**, **104**, or **106**. The link interface in FIG. **10** is for use when switch **108** is a TSI switch. In one implementation, the link interface shown in FIG. **10** is a PCB. FIG. **10** will be described with reference to link interface **100**, but the implementation shown in FIG. **10** can be applicable to other link interface modules. In one implementation, each link interface resides in switch **90** as a PCB.

Link interface **100** includes transceiver **300** for receiving and transmitting data signals on medium **122** in accordance with the physical signaling requirements of medium **122**. In one implementation, transceiver **300** is an optical transceiver. In another embodiment, transceiver **300** is a Giga-bit Ethernet transceiver for exchanging physical signals with medium **122** in accordance with the physical signaling standards of Giga-bit Ethernet.

Transceiver **300** is coupled to Layer 1/Layer 2 processing module **302**. During ingress, transceiver **300** sends signals from medium **122** to processing module **302**. In one implementation, processing module **302** carries out all Layer 1 processing for incoming data and a portion of required Layer 2 processing. In some implementations, processing module **302** does not perform any Layer 2 processing. Processing module **302** supports different protocols in various embodiments. During an egress operation, processing module **302** processes data according to Layer 1 and Layer 2 protocols to prepare the data for transmission onto medium **122**.

Processing module **302** is coupled to slot mapper **303**. During ingress, slot mapper **303** obtains data from processing module **302**. Slot mapper **303** performs the above-described operations for mapping data into virtual channel time slots (steps **51** and **52**, FIG. **3**A) and forwarding time slot to TSI switch **108** (step **14**, FIG. **2**A). During egress, slot mapper **303** receives data from processing engines over switch plane **152**. Slot mapper **303** maps slot data into virtual channels for use by processing module **302** in performing Layer 2 framing and Layer 1 processing.

Slot mapper **303** is coupled to switch plane interface **304**. Switch plane interface **304** is coupled to switch plane **152** to transfer data between channel mapper **303** and plane **152**. During data ingress, switch plane interface **304** forwards sets of time slots from slot mapper **303** onto switch plane **152**. In one implementation, interface **304** sends sets of time slots over switch plane **152** in the form of GFP framed data over SONET. During egress, interface **304** transfers data from switch plane **152** to slot mapper **303**.

Controller **308** directs the operation of transceiver **300**, processing module **302**, slot mapper **303**, and switch plane **304**. Controller **308** is coupled to these components to exchange information and control signals. Controller **308** is also coupled to local memory **306** for accessing data and software instructions that direct the operation of controller **308**. Controller **308** is coupled to control bus interface **310**, which facilitates the transfer of information between link interface **100** and control bus **150**. Controller **308** can be implemented using any standard or proprietary microprocessor or other control engine. Controller **308** responds to instructions from control module **130** that are received via control bus **150**. Memory **307** is coupled to controller **308**, Layer 1/Layer 2 processing module **302**, and slot mapper **303** for maintaining instructions and data.

Controller **308** performs several functions in one embodiment. Controller **308** collects network related statistics generated by transceiver **300** and Layer 1/Layer 2 processing module **302**. Example statistics include carrier losses on medium **122** and overflows in Layerl/Layer 2 processing module **302**. Controller **308** and control module **130** employ these statistics to determine whether any failures have occurred on link interface **100**. The collected statistics can also enable controller **308** and control module **130** to determine the level of bandwidth traffic currently passing through link interface **100**. Control module **130** uses this information to ultimately decide how to distribute the bandwidth capacity of link interfaces and processing engines within switch **90**. Controller **308** carries out instructions from control module **130** when implementing link interface and processing engine switchovers to account for failures or improved resource utilization. The instructions may call for activating or deactivating transceiver **300**.

In one example, controller **308** identifies a failure in transceiver **300**. Controller **308** stores this indication in a database in memory **307**. The failure information stored in memory is provided to control module **130**. Control module **130** uses this information to deactivate link interface **100** and initiate a switchover process—assigning one or more link interfaces in switch **90** to begin carrying out the operations of link interface **100**.

In another example, controller **308** provides control module **130** with information relating to the amount of bandwidth being utilized on link **122**—indicating whether link interface **100** can handle more traffic or needs assistance in handling the current traffic. Based on this information, control module **130** may decide to switchover some of the responsibilities of link interface **100** to one or more different link interfaces. If a switchover is needed, control module **120** arranges for the mapping table information to be modified, as described above for one embodiment.

FIG. **11** is a block diagram depicting an alternate embodiment of a link interface when switch **108** is a packet switch. The components of FIG. **11** that are numbered the same as a component in FIG. **10** operate the same as described for FIG. **10**. The only difference is that slot mapper **303** from FIG. **10** is replaced by packet mapper **309**. Packet mapper **309** is coupled to exchange data with Layer 1\Layer 2 processing module **302** and switch plane interface **304**.

During ingress, packet mapper **309** maps data into packets (step **70**, FIG. **6**). Packet mapper **309** retrieves data from processing module **302**. Packet mapper **309** maps the data into packet payloads and places headers on the packets. Packet mapper **309** then forwards the packets to switch plane **304**, which forwards the packets to packet switch **108**.

During egress, packet mapper **309** assists in generating data frames for transmission (step **87**, FIG. **7**). Packet mapper **309** receives data from switch plane interface **304** in the form of packets formatted for packet switch **108**. Packet mapper **309** places the data for the packets into a format that allows processing module **302** to properly direct the packet payloads into frames for transmission by transceiver **300**.

FIG. **12** is a block diagram depicting one embodiment of a processing engine in switch **90**, such as processing engines **110**, **112**, **114**, and **116**. In one embodiment, processing engines **110**, **112**, **114**, and **116** are each implemented as PCBs in switch **90**. In one implementation, each processing engine in switch **90** support all of the protocols for each OSI model layer supported on the processing engine. This enables any processing engine to exchange data with any link interface in switch **90**. This provides switch **90** with the freedom to allocate processing engine resources without considering the protocol employed in incoming data. The granularity of internal data switching between link interfaces and processing engines can vary in different embodiments. In one embodiment, switch **90** is able to individually switch a single time slot of data from each link interface to a processing engine.

Although FIG. **12** is described with respect to processing engine **110**, the description applies to all processing engines in switch **90**. Processing engine **110** includes network processor **338** coupled to exchange information with fabric plane interface **336** and switch plane interface **342** via conversion engine **335**. Interface **342** is coupled to switch plane **152** to exchange data between processing engine **110** and switch **108**. Interface **336** is coupled to fabric plane **154**. Interface **336** uses plane **154** to exchange data between processing engine **110** and fabric **120**.

During data ingress, interface **342** receives data provided on plane **152**. Interface **342** provides the data to conversion engine **335**. Conversion engine **335** extracts payloads (step **20**, FIG. **2**A) from received sets of time slots for processing (step **22**, FIG. **2**A) at Layer 2 and above. Conversion engine **335** maps an extracted payload into a desired packet format and forwards the packet to network processor **338** for processing.

Network processor **338** processes data from plane **152** according Layer 2 protocols and above. Network processor **338** also performs the above-described function of generating fabric cells (step **24**, FIG. **2**A). Fabric plane interface **336** receives fabric cells from network processor **338**. Interface **336** transmits the fabric payload onto fabric plane **154** (step **26**, FIG. **2**A).

During data egress, network processor **338** processes data in fabric cells received from fabric plane **154** through fabric plane interface **336**—reassembling cells into packets and processing the packets at Layer 2 and above (steps **32** and **34**, FIG. **2**B). Network processor **338** passes processed data to conversion engine **335**. Conversion engine **335** maps the data into one or more virtual channel time slots (step **36**, FIG. **2**B). Conversion engine **335** passes egress sets of time slots to plane **152** via switch plane interface **342**. Interface **342** places sets of time slots on plane **152**, which carries the data to switch **108**.

In an alternate embodiment, switch **108** is a packet switch. In this embodiment, conversion engine **335** converts data between processing packets and packets exchanged with packet switch **108** (step **77**, FIG. **6** and step **80**, FIG. **7**).

Network processor **338** carries out operations that support the applications running on switch **90**. For example, switch **90** may support virtual private networks by acting as a Provider Edge Router. Network processor **338** maintains routing tables for the virtual private networks. Processing engine **338** employs the tables to properly route data for a VPN to the next step in a virtual circuit in the VPN.

Processing engine **110** also includes controller **332**, which is coupled to local memory **334** and control bus interface **330**. Network processor **338** is coupled to controller **332** to receive data and control instructions. Controller **332** performs many of the same functions described above for controller **308** on link interface **100**, except that controller **332** performs operations specific to the operation of processing engine **110**. Local memory **334** holds instructions for controller **332** to execute, as well as data maintained by controller **332** when operating. Control bus interface **330** operates the same as the above-described control bus interface **310** in FIG. **10**. Memory **333** is coupled to controller **332** and network processor **338** to maintain data and instructions.

One application performed on controller **332** is the maintenance of network related statistics. Network processor **338** collects statistics based on information in the data frames passing through processing engine **110**. These statistics identify whether a failure has occurred on processing engine **110** or another component within switch **90**. Additional statistics collected by network processor **338** indicate the level of utilization that processing engine **110** is experiencing. These statistics are made available to controller **332** for delivery to control module **130**.

Example statistics include whether frames have been dropped and the number of frames passing through network processor **338**. When a failure is detected, controller **332** signals control module **130** over bus **150**. In one implementation, controller **332** performs this operation by sending data over bus **150** that contains information to indicate that a failure has taken place. Similarly controller **332** can send information over bus **150** to control module **130** that indicates the level of bandwidth utilization on processing engine **110**. Alternatively, control module **130** can access raw statistics in local memory **334** and memory **333** and make failure and utilization assessments.

In response to the statistics provided by controller **332**, control module **130** may decide that it is appropriate to perform a switchover that involves processing engine **110** or other components within switch **90**. Control module **130** sends instructions to controller **332** over bus **150** to identify the actions for processing engine **110** to implement to facilitate a switchover. These actions may include activating or deactivating processing engine **110**. In the case of processing engine **110** being substituted for another processing engine, control module **130** may provide controller **332** with information that brings processing engine **110** to the current state of the other processing engine. This allows processing engine **110** to operate in place of the replaced component.

Controller **332** can also support the performance of many other applications by network processor **338**. In various embodiments, controller **332** can direct the operation of network processor **338** in performing tunneling, frame relay support, and Ethernet switching and bridging functions. These are only examples of some applications that can be performed on processing engine **110**. A wide variety of applications can operate on processing engine **110**.

FIG. **13** is a block diagram depicting one embodiment of a single line card module that contains both fabric **120** and switch **108**. Switch **108** directs data between link interfaces and processing engines over switch plane **152**. During ingress, data passes from an ingress link interface onto plane **152**, into switch **108**, back onto plane **152**, and into one or more processing engines. During egress, switch **108** receives data from an egress processing engine on plane **152** and provides that data to one or more egress link interfaces via plane **152**. Fabric **120** provides for the exchange of data between processing engines. Fabric **120** receives data on plane **154** from an ingress processing engine and passes the data to an egress processing engine on plane **154**.

Switch **108** and fabric **120** are both coupled to controller **366**. Controller **366** interfaces with local memory **368** and network control bus interface **364** in a manner similar to the one described above for controller **308** in link interface **100** (FIG. **10**). Memory **368** maintains instructions for directing the operation of controller **366**, as well as data employed by controller **366** in operation. Control bus interface **364** allows controller **366** to exchange data and control information with control module **130** over control bus **150**. In one implementation, control bus interface **364** supports the transmission of 100 Base-T Ethernet information over control bus **150**.

As with the controllers described above, controller **366** supports the performance of a number of applications by fabric **120** and switch **108**. In one application, controller **366** collects statistical information from switch **108** and fabric **120**. One type of statistical information identifies the amount of data passing through fabric **120** and switch **108**. Other statistics indicate whether switch **108** or fabric **120** have failed. Those skilled in the art will recognize that various embodiments of the invention allow for controller **366** to collect a wide array of different statistical information. Controller **366** communicates the collected statistical information to control module **130** over bus **150**. Control module **130** uses the statistical information to determine whether the responsibilities assigned to any link interface or processing engine need to be redistributed.

Controller **366** also supports the redistribution of responsibilities—enabling control module **130** to change switching rules in switch **108**. For example, controller **366** can program the above-described Backup field values in TSI switch **108**—redistributing time slot data among different link interfaces and processing engines.

FIG. **14** is a block diagram depicting one embodiment of TSI switch **108**. TSI switch **108** includes an incoming TSI switch port for each link interface and an incoming TSI switch port for each processing engine. Each incoming TSI switch port is coupled to either a link interface or processing engine. In one embodiment, TSI switch **108** includes 24 incoming TSI switch ports coupled to link interfaces and 12 incoming TSI switch ports coupled to processing engines. A subset of the incoming TSI switch ports in TSI switch **108** are shown in FIG. **14** as TSI switch ports **380**, **382** and **384**. Incoming TSI switch ports coupled to link interfaces receive ingress data in the form of a set of time slots, such as 48 time slots sent in the format of GFP framed data over a SONET. Incoming TSI switch ports coupled to processing engines receive egress data in the form of a set of time slots, such as 48 time slots sent in the format of GFP framed data over SONET.

The incoming TSI switch ports are used during the process steps described above with reference to FIG. **4** for receiving and mapping incoming time slot data to outgoing time slots. Each incoming TSI switch port is coupled to switch plane **152** to receive a set of time slots from either a link interface or processing engine. Each incoming TSI switch port is also coupled to memory interface **400**. When an incoming TSI switch port receives a time slot of data (step **60**, FIG. **4**), TSI switch **108** maps the slot data to a time slot in an outgoing set of time slots (step **62**, FIG. **4**). TSI switch **108** maps the slot data by storing it into a location in memory **404** that is designated for the slot in the outgoing set of time slots. Each incoming TSI switch port is coupled to memory interface **400**, which is coupled to memory bus **406**. Memory bus **406** is coupled to memory **404** to exchange data. In operation, data from a slot in an incoming TSI switch port is provided to memory interface **400** along with an identifier for a slot in an outgoing set of time slots. Memory interface **400** loads the data from the incoming TSI switch port's slot into a location in memory **404** that corresponds to the identified time slot in the outgoing set of time slots.

TSI switch **108** also includes connection control **396**. Connection control **396** is coupled to memory interface **400** to provide mapping information. The information from connection control **396** informs memory interface **400** where to map each incoming time slot. In one implementation, connection control **396** includes the above-described mapping tables employed by TSI switch **108**.

TSI switch **108** also includes a set of outgoing TSI switch ports. Each outgoing TSI switch port is coupled to either a link interface or a processing engine to forward outgoing sets of time slots. TSI switch **108** includes an outgoing TSI switch port for each link interface and an outgoing TSI switch port for each processing engine. The outgoing TSI switch ports are coupled to the link interfaces and processing engines over switch plane **152**. Outgoing TSI switch ports coupled to processing engines deliver outgoing sets of time slots to the processing engine during ingress data flow. Outgoing TSI switch ports coupled to link interfaces provide outgoing sets of time slots to the link interfaces during egress data flow. FIG. **14** shows a subset of the outgoing TSI switch ports as transmit ports **386**, **388** and **390**.

The outgoing TSI switch ports are used in carrying out the forwarding of outgoing sets of time slots as shown above in FIG. **5**. When an outgoing set of time slots needs to be transmitted, memory interface **402** retrieves the data for the time slots from locations in memory **404** that are designated to the slots (step **66**, FIG. **5**). Connection control **396** is coupled to memory interface **402** to indicate whether valid data exists in memory **404** for a time slot or idle data needs to be resident in the portion of the outgoing TSI switch port corresponding to the slot. When valid data exists, memory interface **402** retrieves the data from memory **404**.

Each outgoing TSI switch port communicates with memory **404** through memory interface **402** over memory bus **406**. Each outgoing TSI switch port is coupled to memory interface **402**. Memory interface **402** is coupled to memory bus **406** to retrieve data from memory **404** to service channel data requests from transmit ports.

In alternate embodiments, different designs can be employed for TSI switch **108** that facilitate the above-described operation of TSI switch **108**. In various embodiments, different TDM switches can be employed.

The foregoing detailed description of the invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. The described embodiments were chosen in order to best explain the principles of the invention and its practical application to thereby enable others skilled in the art to best utilize the invention in various embodiments and with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the claims appended hereto.

We claim:

1. A network switch comprising:
a plurality of link interfaces;
a plurality of processing engines;
a switch fabric coupled to said plurality of processing engines; and
a switch coupling said plurality of link interfaces to said plurality of processing engines, wherein said switch is configured to:
map data from at least one time slot in a set of time slots from a link interface in said plurality of link interfaces to at least one outgoing set of time slots; and
forward each outgoing set of time slots in said at least one outgoing set of time slots to a processing engine in said plurality of processing engines,
wherein said switch is configured to map data in response to mapping information maintained in said network switch, wherein said mapping information identifies a time slot in an outgoing set of time slots in said at least one outgoing set of time slots for each time slot in said set of time slots; and
said network switch is configured to modify said mapping information in response to a failure of at least one link interface in said plurality of link interfaces.

2. The network switch according to claim 1, wherein said mapping information includes entries in a mapping table.

3. A network switch comprising:
a plurality of link interfaces;
a plurality of processing engines;
a switch fabric coupled to said plurality of processing engines; and
a switch coupling said plurality of link interfaces to said plurality of processing engines, wherein said switch is configured to:
map data from at least one time slot in a set of time slots from a processing engine in said plurality of processing engines to at least one outgoing set of time slots; and
forward each outgoing set of time slots in said at least one outgoing set of time slots to a link interface in said plurality of link interfaces,
wherein said switch is configured to map data in response to mapping information maintained in said network switch, wherein said mapping information identifies a time slot in an outgoing set of time slots in said at least one outgoing set of time slots for each time slot in said set of time slots; and
said network switch is configured to modify said mapping information in response to a failure of at least one link interface in said plurality of link interfaces.

4. The network switch according to claim 3, wherein said mapping information includes entries in a mapping table.

5. A network switch comprising:
a plurality of link interfaces;
a plurality of processing engines;
a switch fabric coupled to said plurality of processing engines; and

a switch coupling said plurality of link interfaces to said plurality of processing engines, wherein switch is configured to:
map data from a first link interface in said plurality of link interfaces to multiple processing engines in said plurality of processing engines; and
map data from multiple link interfaces in said plurality of link interfaces to a first processing engine in said plurality of processing engines,
wherein said switch is configured to map data from a first link interface and map data from multiple link interfaces in response to mapping information maintained in said network switch, and
said network switch is configured to modify said mapping information in response to a failure of at least one link interface in said plurality of link interfaces;
wherein at least one processing engine in said plurality of processing engines receives data to be processed by said at least one processing engine according to a first protocol within a layer and data to be processed by said at least one processing engine according to a second protocol within said layer and said first protocol is different than said second protocol.

6. The network switch according to claim 5, wherein said mapping information includes entries in a mapping table.

7. The network switch according to claim 6, wherein said switch is configured to modify said mapping information including by:
(1) modifying at least one Backup field value in said mapping table.

8. The network switch according to claim 5, wherein said switch is a switch from a group of switches consisting of a packet switch, a multiplexing switch, and a time slot interchange switch.

9. The network switch according to claim 5, wherein:
said plurality of link interfaces includes at least twice as many link interfaces as a number of processing engines included in said plurality of processing engines;
each link interface in said set of link interfaces has redundancy;
each processing engine in said set of processing engines has redundancy; and
no processing engine in said set of processing engines is idle.

10. A network switch comprising:
a plurality of link interfaces;
a plurality of processing engines;
a switch fabric coupled to said plurality of processing engines; and
a switch coupling said plurality of link interfaces to said plurality of processing engines, wherein said switch is configured to:
map data from at least one time slot in a set of time slots from a link interface in said plurality of link interfaces to at least one outgoing set of time slots; and
forward each outgoing set of time slots in said at least one outgoing set of time slots to a processing engine in said plurality of processing engines,
wherein said switch is configured to map data in response to mapping information maintained in said network switch, wherein said mapping information identifies a time slot in an outgoing set of time slots in said at least one outgoing set of time slots for each time slot in said set of time slots; and
said network switch is configured to modify said mapping information in response to a failure of at least one processing engine in said plurality of processing engines.

11. The network switch according to claim **10**, wherein said mapping information includes entries in a mapping table.

12. A network switch comprising:

a plurality of link interfaces;

a plurality of processing engines;

a switch fabric coupled to said plurality of processing engines; and

a switch coupling said plurality of link interfaces to said plurality of processing engines, wherein said switch is configured to:

map data from at least one time slot in a set of time slots from a processing engine in said plurality of processing engines to at least one outgoing set of time slots; and

forward each outgoing set of time slots in said at least one outgoing set of time slots to a link interface in said plurality of link interfaces,

wherein said switch is configured to map data in response to mapping information maintained in said network switch, wherein said mapping information identifies a time slot in an outgoing set of time slots in said at least one outgoing set of time slots for each time slot in said set of time slots; and

said network switch is configured to modify said mapping information in response to a failure of at least one processing engine in said plurality of processing engines.

13. The network switch according to claim **12**, wherein said mapping information includes entries in a mapping table.

14. A network switch comprising:

a plurality of link interfaces;

a plurality of processing engines;

a switch fabric coupled to said plurality of processing engines; and

a switch coupling said plurality of link interfaces to said plurality of processing engines, wherein said switch is configured to:

map data from a first link interface in said plurality of link interfaces to multiple processing engines in said plurality of processing engines; and

map data from multiple link interfaces in said plurality of link interfaces to a first processing engine in said plurality of processing engines,

wherein said switch is configured to map data from a first link interface and map data from multiple link interfaces in response to mapping information maintained in said network switch; and

said network switch is configured to modify said mapping information in response to a failure of at least one processing engine in said plurality of processing engines;

wherein at least one processing engine in said plurality of processing engines receives data to be processed by said at least one processing engine according to a first protocol within a layer and data to be processed by said at least one processing engine according to a second protocol within said layer and said first protocol is different than said second protocol.

15. The network switch according to claim **14**, wherein said mapping information includes entries in a mapping table.

16. The network switch according to claim **15**, wherein said switch is configured to modify said mapping information including by:

(1) modifying at least one Backup field value in said mapping table.

17. The network switch according to claim **14**, wherein said switch is a switch from a group of switches consisting of a packet switch, a multiplexing switch, and a time slot interchange switch.

18. The network switch according to claim **14**, wherein:

said plurality of link interfaces includes at least twice as many link interfaces as a number of processing engines included in said plurality of processing engines;

each link interface in said set of link interfaces has redundancy;

each processing engine in said set of processing engines has redundancy; and

no processing engine in said set of processing engines is idle.

19. A network switch comprising:

a plurality of link interfaces;

a plurality of processing engines;

a switch fabric coupled to said plurality of processing engines; and

a switch coupling said plurality of link interfaces to said plurality of processing engines, wherein said switch is configured to:

map data from a first link interface in said plurality of link interfaces to multiple processing engines in said plurality of processing engines;

map data from multiple link interfaces in said plurality of link interfaces to a first processing engine in said plurality of processing engines,

wherein said switch is configured to map data from a first link interface and map data from multiple link interfaces

in response to mapping information maintained in said network switch; and

said network switch is configured to modify said mapping information, including modifying at least one Backup field value in said mapping table;

wherein at least one processing engine in said plurality of processing engines receives data to be processed by said at least one processing engine according to a first protocol within a layer and data to be processed by said at least one processing engine according to a second protocol within said layer and said first protocol is different than said second protocol.

20. The network switch according to claim **19**, wherein said switch is configured to modify said mapping information in response to a utilization level of at least one link interface in said plurality of link interfaces.

21. The network switch according to claim **19**, wherein said switch is configured to modify said mapping information in response to a failure of at least one link interface in said plurality of link interfaces.

22. The network switch according to claim **19**, wherein said switch is configured to modify said mapping information in response to a utilization level of at least one processing engine in said plurality of processing engines.

23. The network switch according to claim **19**, wherein said switch is configured to modify said mapping information in response to a failure of at least one processing engine in said plurality of processing engines.

24. The network switch according to claim **19**, wherein said mapping information includes entries in a mapping table.

25. The network switch according to claim **19**, wherein said switch is a switch from a group of switches consisting of a packet switch, a multiplexing switch, and a time slot interchange switch.

26. The network switch according to claim **19**, wherein:

said plurality of link interfaces includes at least twice as many link interfaces as a number of processing engines included in said plurality of processing engines;

each link interface in said set of link interfaces has redundancy;

each processing engine in said set of processing engines has redundancy; and

no processing engine in said set of processing engines is idle.

27. A network switch comprising:

a plurality of link interfaces;

a plurality of processing engines;

a switch fabric coupled to said plurality of processing engines; and

a switch coupling said plurality of link interfaces to said plurality of processing engines, wherein said switch is configured to:

map data from a first link interface in said plurality of link interfaces to multiple processing engines in said plurality of processing engines; and

map data from multiple link interfaces in said plurality of link interfaces to a first processing engine in said plurality of processing engines;

wherein said switch is configured to map data from a first link interface and map data from multiple link interfaces in response to mapping information maintained in said network switch; and

wherein at least one processing engine in said plurality of processing engines receives data to be processed by said at least one processing engine according to a first protocol within a layer and data to be processed by said at least one processing engine according to a second protocol within said layer and said first protocol is different than said second protocol; and

wherein:

said plurality of link interfaces includes at least twice as many link interfaces as a number of processing engines included in said plurality of processing engines;

each link interface in said set of link interfaces has redundancy;

each processing engine in said set of processing engines has redundancy; and

no processing engine in said set of processing engines is idle.

28. The network switch according to claim 27, wherein said method includes the step of:

(c) modifying said mapping information.

29. The network switch according to claim 28, wherein said switch is configured to modify said mapping information in response to a utilization level of at least one link interface in said plurality of link interfaces.

30. The network switch according to claim 28, wherein said switch is configured to modify said mapping information in response to a failure of at least one link interface in said plurality of link interfaces.

31. The network switch according to claim 28, wherein said switch is configured to modify said mapping information in response to a utilization level of at least one processing engine in said plurality of processing engines.

32. The network switch according to claim 28, wherein said switch is configured to modify said mapping information in response to a failure of at least one processing engine in said plurality of processing engines.

33. The network switch according to claim 28, wherein said mapping information includes entries in a mapping table.

34. The network switch according to claim 28, wherein said switch is configured to modify said mapping information including by:

(1) modifying at least one Backup field value in said mapping table.

35. The network switch according to claim 27, wherein said switch is a switch from a group of switches consisting of a packet switch, a multiplexing switch, and a time slot interchange switch.

36. The network switch according to claim 5, wherein said first protocol is High-level Data Link Control ("HDLC"), Point-to-Point Protocol ("PPP"), Frame Relay, Asynchronous Transfer Mode ("ATM"), Gigabit Ethernet, or Internet Protocol ("IP"); and said second protocol is High-level Data Link Control ("HDLC"), Point-to-Point Protocol ("PPP"), Frame Relay, Asynchronous Transfer Mode ("ATM"), Gigabit Ethernet, or Internet Protocol ("IP").

37. The network switch according to claim 14, wherein said first protocol is High-level Data Link Control ("HDLC"), Point-to-Point Protocol ("PPP"), Frame Relay, Asynchronous Transfer Mode ("ATM"), Gigabit Ethernet, or Internet Protocol ("IP"); and said second protocol is High-level Data Link Control ("HDLC"), Point-to-Point Protocol ("PPP"), Frame Relay, Asynchronous Transfer Mode ("ATM"), Gigabit Ethernet, or Internet Protocol ("IP").

38. The network switch according to claim 27, wherein said first protocol is High-level Data Link Control ("HDLC"), Point-to-Point Protocol ("PPP"), Frame Relay, Asynchronous Transfer Mode ("ATM"), Gigabit Ethernet, or Internet Protocol ("IP"); and said second protocol is High-level Data Link Control ("HDLC"), Point-to-Point Protocol ("PPP"), Frame Relay, Asynchronous Transfer Mode ("ATM"), Gigabit Ethernet, or Internet Protocol ("IP").

* * * * *

Exhibit B

US007609621B1

(12) **United States Patent**
Kanagala et al.

(10) **Patent No.: US 7,609,621 B1**
(45) **Date of Patent: Oct. 27, 2009**

(54) AUTOMATIC PROTECTION NETWORK SWITCHING

(76) Inventors: **Sameer Kanagala**, 640 Clyde Ct., Mountain View, CA (US) 94043; **Jan Medved**, 640 Clyde Ct., Mountain View, CA (US) 94043; **Alex Dadnam**, 640 Clyde Ct., Mountain View, CA (US) 94043

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 717 days.

(21) Appl. No.: **10/776,460**

(22) Filed: **Feb. 10, 2004**

(51) **Int. Cl.**
G01R 31/04 (2006.01)

(52) **U.S. Cl.** **370/227**; 370/228

(58) **Field of Classification Search** 370/225–228
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,193,086 A *	3/1993	Satomi et al.	....	370/228
5,216,666 A *	6/1993	Stalick	....	370/222
6,195,330 B1 *	2/2001	Sawey et al.	....	370/220
6,196,330 B1 *	3/2001	Matthias et al.	....	173/48
6,332,198 B1 *	12/2001	Simons et al.	....	714/6
6,574,477 B1 *	6/2003	Rathunde	....	455/453
6,868,057 B1 *	3/2005	Sha	....	370/216
2003/0117952 A1 *	6/2003	Ueno et al.	....	370/228
2003/0165115 A1 *	9/2003	Sutoh et al.	....	370/216
2004/0086003 A1 *	5/2004	Natarajan et al.	....	370/545

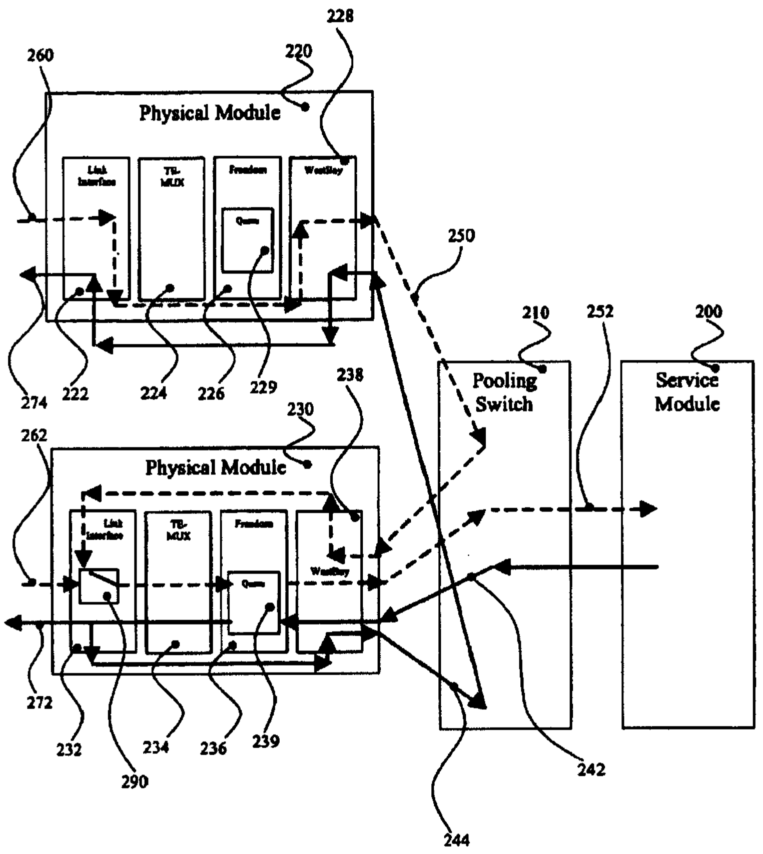
* cited by examiner

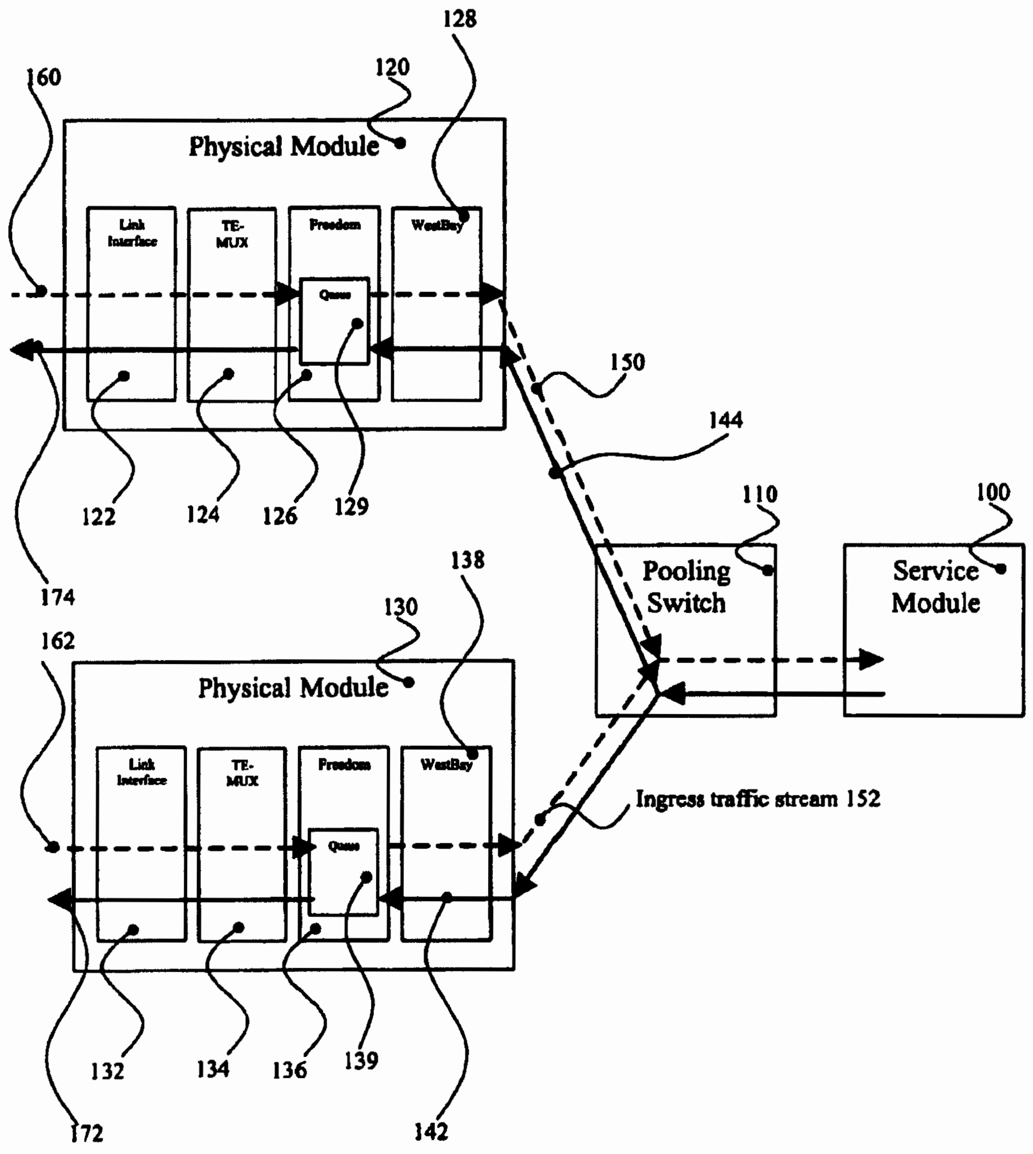
Primary Examiner—Kwang B Yao
Assistant Examiner—Andrew Lai

(57) **ABSTRACT**

A method of protecting a protected link is disclosed. The method includes connecting traffic from a service module to a first physical module having a link layer framer that is connected to a protected egress link. The traffic is connected through the first physical module through a pooling switch to a second physical module that is connected to an alternate egress link.

66 Claims, 6 Drawing Sheets





128
120
160
Physical Module
Link Interface
TE-MUX
Freedom
WestBay
Queue
150
144
122
124
126
129
110
100
Pooling Switch
Service Module
174
138
130
162
Physical Module
Link Interface
TE-MUX
Freedom
WestBay
Queue
Ingress traffic stream 152
132
134
136
139
172
142

FIG. 1

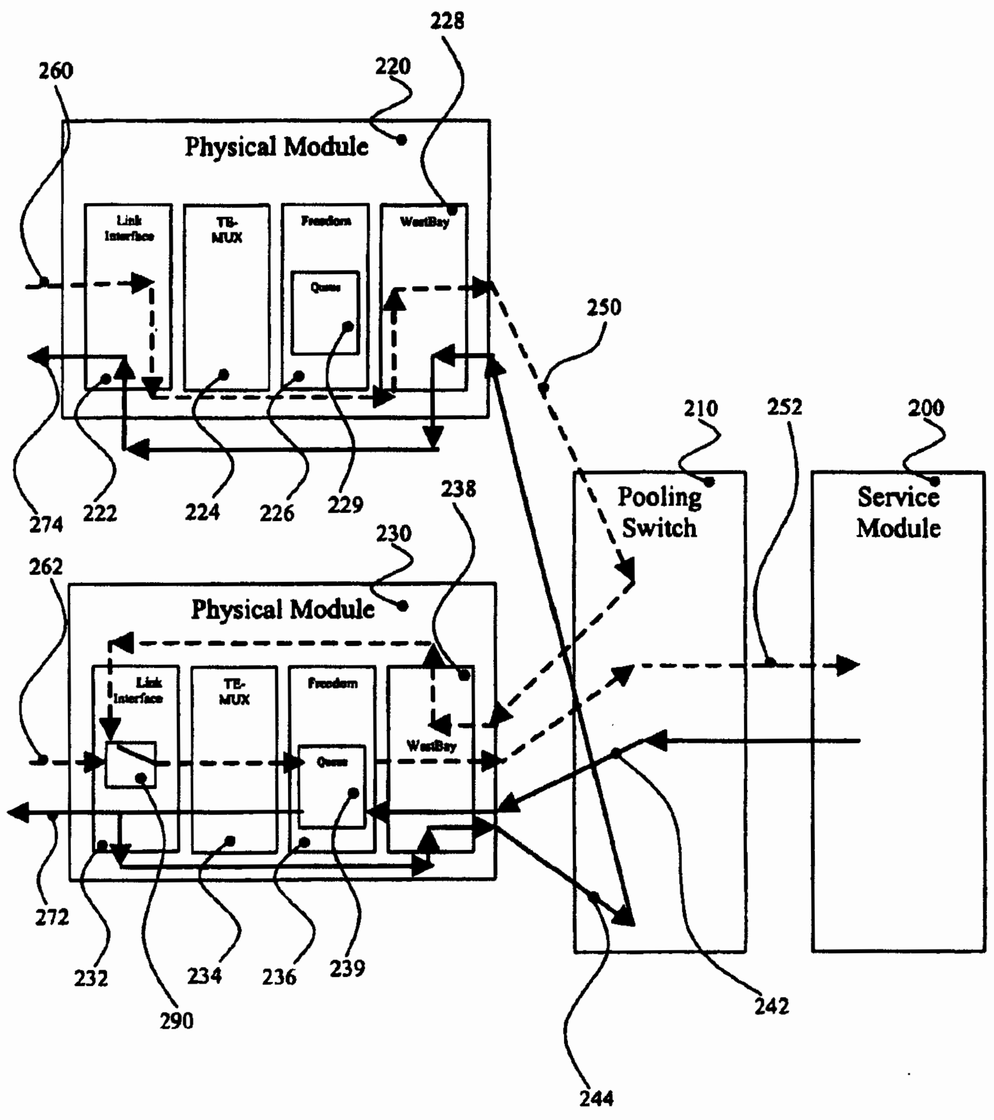

228
260
220
Physical Module
Link Interface
TE-MUX
Freedom
Queue
WestBay
250
210
252
200
Pooling Switch
Service Module
274
222
224
226
229
238
230
262
Physical Module
Link Interface
TE-MUX
Freedom
Queue
WestBay
272
232
234
236
239
290
242
244

FIG. 2

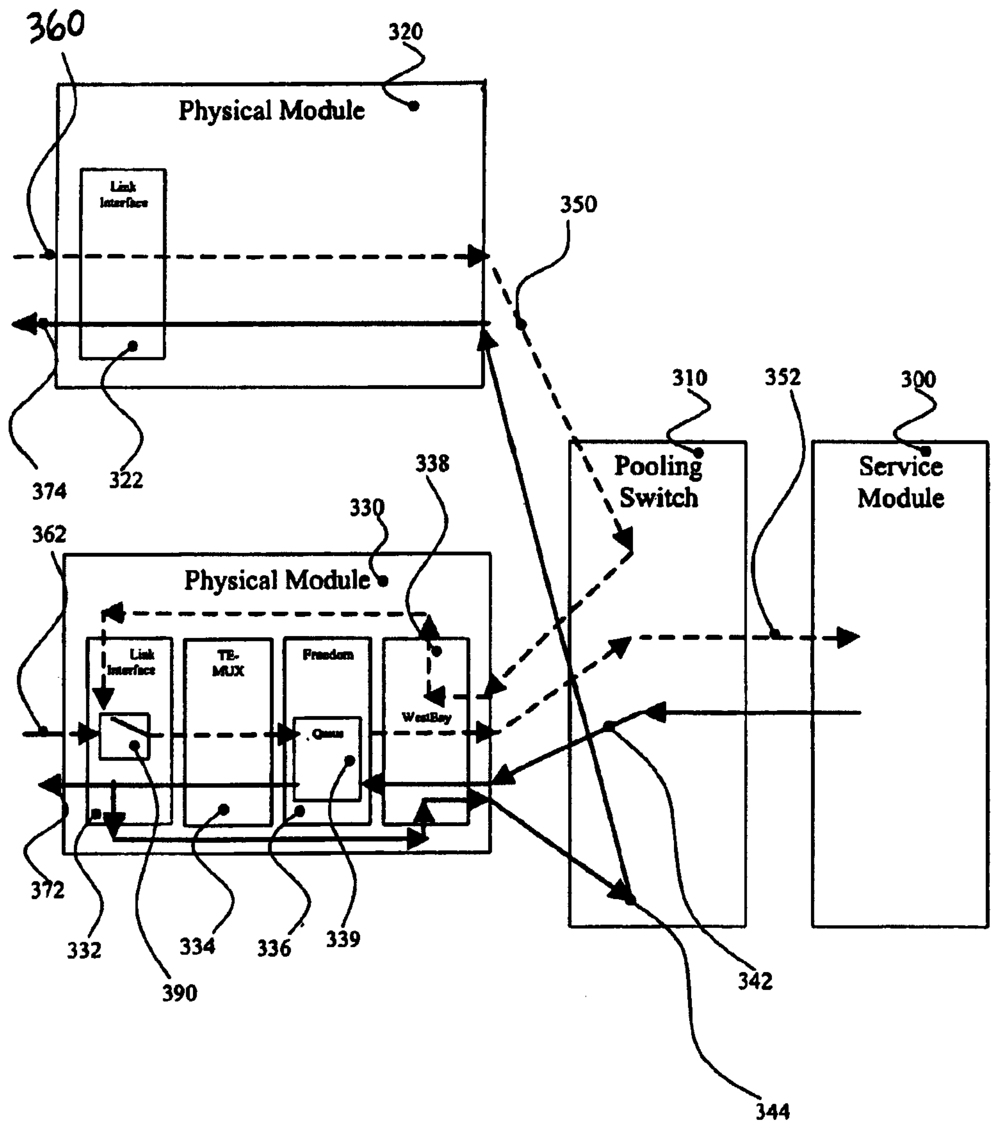

360
320
Physical Module
Link Interface
350
374
322
310
352
300
Pooling Switch
Service Module
338
330
362
Physical Module
Link Interface
TE-MUX
Freedom
Queue
WestBay
372
332
334
336
339
390
342
344

FIG. 3

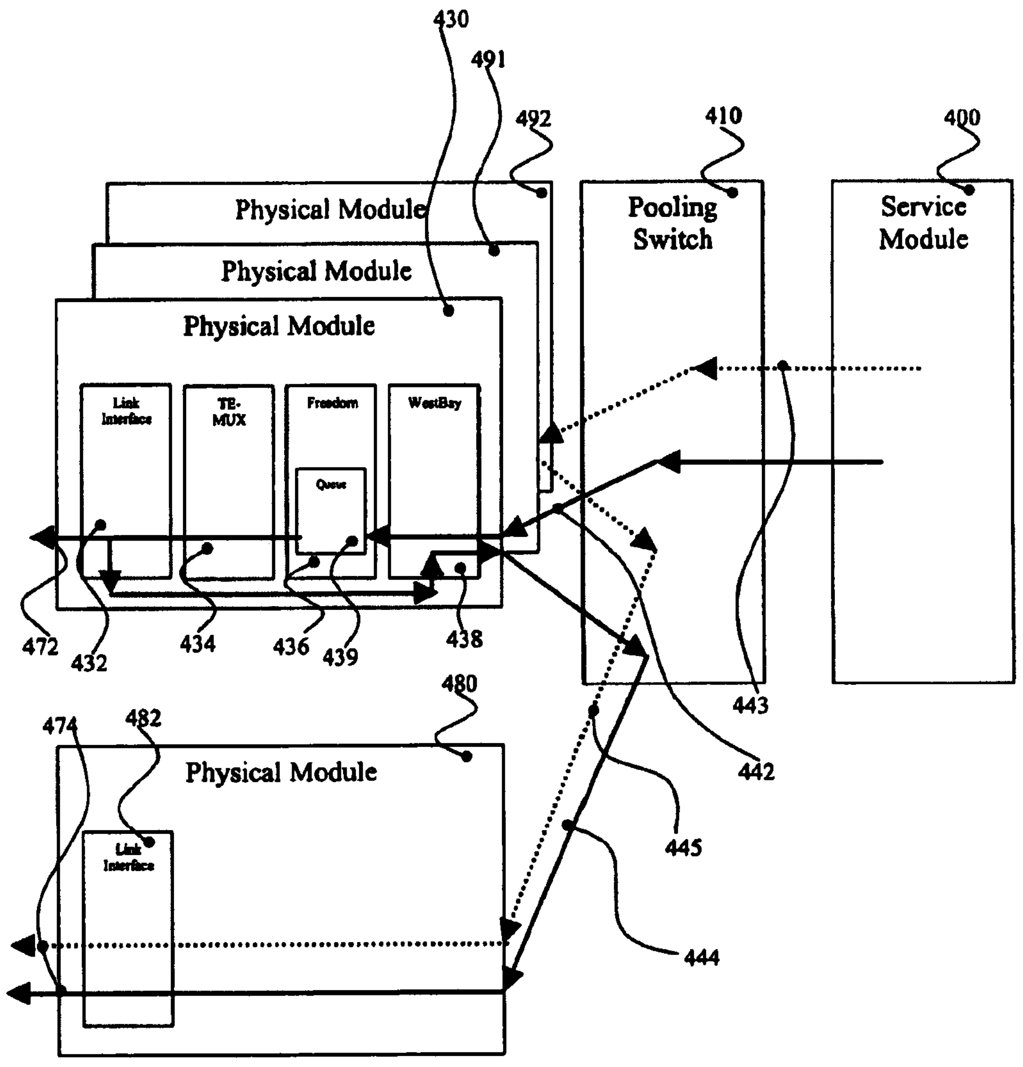

430
491
492
410
400
Physical Module
Physical Module
Physical Module
Pooling Switch
Service Module
Link Interface
TE-MUX
Freedom
WestBay
Queue
472
432
434
436
439
438
443
442
480
474
482
Physical Module
Link Interface
445
444

FIG. 4

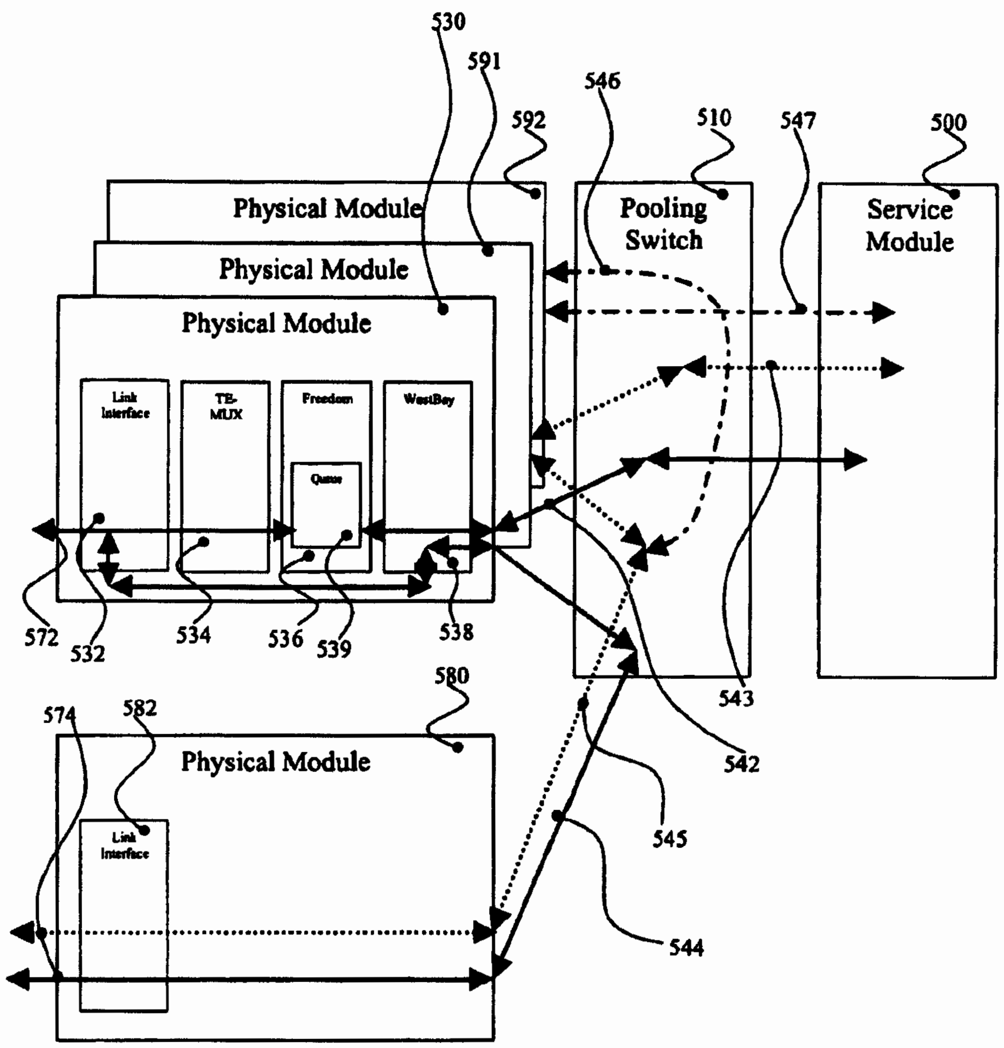

530
591
592
546
510
547
500
Physical Module
Physical Module
Physical Module
Pooling Switch
Service Module
Link Interface
TE-MUX
Freedom
WestBay
Queue
572
532
534
536
539
538
543
542
580
574
582
Physical Module
Link Interface
545
544

FIG. 5

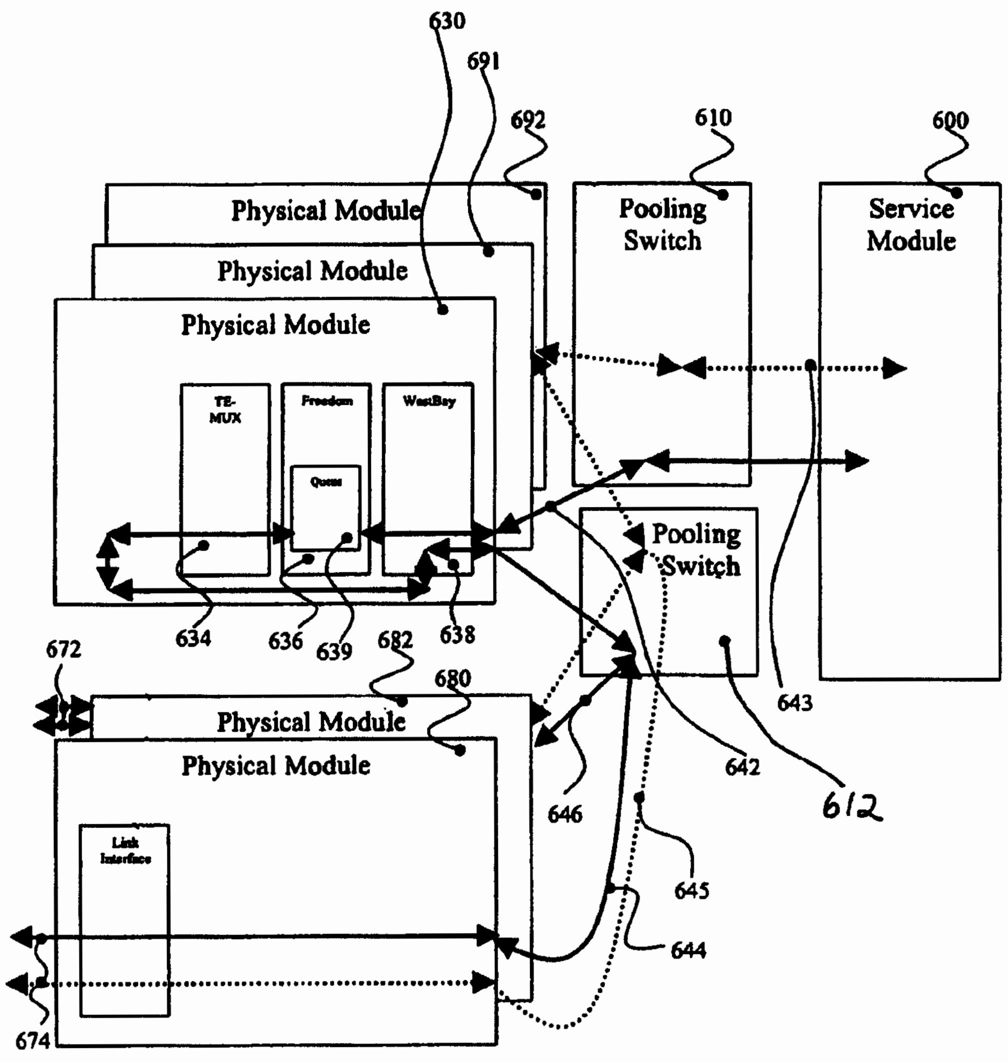

630
691
692
610
600
Physical Module
Physical Module
Physical Module
Pooling Switch
Service Module
TE-MUX
Freedom
WestBay
Queue
Pooling Switch
634
636
639
682
638
672
680
Physical Module
Physical Module
Link Interface
643
642
612
646
645
644
674

FIG. 6

AUTOMATIC PROTECTION NETWORK SWITCHING

FIELD OF THE INVENTION

The present invention relates generally to network switching. More specifically, automatic protection switching is disclosed.

BACKGROUND OF THE INVENTION

Protection of data traffic is an increasingly important technical requirement. For example, it is important to businesses that their network connections do not fail as more and more commerce is conducted using electronic communication systems. Currently, automatic protection systems for network connections utilize data traffic channels each with their own queues. These queues can be large and are not synchronized between the protected traffic channel and the protecting traffic channel. Thus, if the protected traffic channel fails and the protecting traffic channel takes over, there can be either a loss or repeat of traffic transmitted or received over the channel. These losses or repeats of traffic are not handled well by the network and often fall outside the specifications of the network as a whole. It would be helpful if automatic protection systems could be engineered to provide better protection of traffic within the capabilities and specifications of the network.

BRIEF DESCRIPTION OF THE DRAWINGS

Various embodiments of the invention are disclosed in the following detailed description and the accompanying drawings.

FIG. **1** shows a block diagram of a network switching system.

FIG. **2** shows one embodiment of a network switch that does not have a large synchronization difference between two egress/ingress traffic streams.

FIG. **3** shows a reduced cost embodiment of a network switch that does not have a large synchronization difference between two egress/ingress traffic streams.

FIG. **4** shows a further cost reduced embodiment of a network switch that does not have a large synchronization difference between multiple egress traffic streams.

FIG. **5** shows another embodiment of a network switch.

FIG. **6** is a block diagram illustrating another embodiment of an APS using a pooling switch.

DETAILED DESCRIPTION

The invention can be implemented in numerous ways, including as a process, an apparatus, a system, a composition of matter, a computer readable medium such as a computer readable storage medium or a computer network wherein program instructions are sent over optical or electronic communication links. In this specification, these implementations, or any other form that the invention may take, may be referred to as techniques. In general, the order of the steps of disclosed processes may be altered within the scope of the invention.

A detailed description of one or more embodiments of the invention is provided below along with accompanying figures that illustrate the principles of the invention. The invention is described in connection with such embodiments, but the invention is not limited to any embodiment. The scope of the invention is limited only by the claims and the invention

encompasses numerous alternatives, modifications and equivalents. Numerous specific details are set forth in the following description in order to provide a thorough understanding of the invention. These details are provided for the purpose of example and invention may be practiced according to the claims without some or all of these specific details. For the purpose of clarity, technical material that is known in the technical fields related to the invention has not been described in detail so that the invention is not unnecessarily obscured.

FIG. **1** shows a block diagram of a network switching system. The system includes physical modules **120** and **130**. An input traffic stream coming from an ingress link **160** through physical module **120** is directed through another physical module to an egress link. For example, the output traffic stream could be switched through physical module **120** to egress link **174**.

In some embodiments, a network switching system with an automatic protection system for its ingress traffic has two simultaneous links that can be used to protect the incoming traffic streams in the event that one of the ingress links fails. In the embodiment shown in FIG. **1**, the system takes two input traffic streams: the protected one (ingress traffic stream **150**) coming from ingress link **160** through physical module **120** and the protecting one (ingress traffic stream **152**) coming from ingress link **162** through physical module **130**. If the system is operating normally without failure, the traffic stream from the protected stream is switched to output through another physical module to an egress link. For example, the output traffic stream could go through physical module **120** to egress link **174** along egress traffic stream **144**. If the protected traffic stream fails, then the protecting stream is used as the input traffic stream and switched to the other physical module and output across an egress link.

Pooling switch **110** selects whether the protected traffic stream or the protecting traffic stream is to be passed on to the service module **100**. The pooling switch can be a packet switch, a multiplexing switch, a time division multiplexing switch, or any other switch capable of directing multiple input traffic streams. The service module decides from information within the input traffic stream where to output the traffic stream and instructs the pooling switch to make the appropriate connections. In some embodiments, the connections are static and are done by provisioning. In some embodiments, the service module decision is based on the International Standard Organization's Open System Interconnect (ISO/OSI) layer 2 information that is found in the traffic stream. For example, the information may be an Ethernet address or a media access control (MAC) address.

In some embodiments, a network switching system with an automatic protection system for its egress traffic has two simultaneous links that can be used to protect the outgoing traffic streams in the event that one of the egress links fails. In the embodiment shown in FIG. **1**, the system has two output traffic streams: the protected one (egress traffic stream **144**) going to egress link **174** through physical module **120** and the protecting one (egress traffic stream **142**) going to egress link **172** through physical module **130**. If the system is operating normally without failure, the traffic stream from an input traffic stream (for example, ingress traffic stream **150**) is switched to both the protected and protecting output traffic streams.

Pooling switch **110** selects whether the traffic stream from service module **100** is passed on to the protected output traffic stream or the protecting output traffic stream. The pooling switch can be a packet switch, a multiplexing switch, a time division multiplexing switch, or any other switch capable of directing multiple output traffic streams.

In some embodiments, submodules are included within the physical modules. For example, in the embodiment shown, ingress traffic stream **150** from ingress link **160** first passes through a link interface module **122**. Link interface module **122** is an optical or electrical link interface module used to connect to a type of link such as a OC-198, OC-48, OC-12, OC-3, 10/100 ethernet, gigabit Ethernet, DS1, or DS3 or other link. The ingress traffic stream **150** then passes through a TE-MUX **124** module. This TE-MUX module is a multiplexer/demultiplexer of ISO/OSI layer 1 protocols (e.g. DS1 into DS3, E1 into SONET, E1 into TU2, DS3 into SONET frames, or OC-3 SONET frames into OC-48 frames). The ingress traffic stream **150** then passes through a Freedom **126** module. This Freedom module is a link layer processor. This module may, for example, frame the traffic stream for the High-level Data Link Control (HDLC), circuit emulation, or Asynchronous Transfer Mode (ATM) based protocols. Within this module, as part of the link layer framing, ingress traffic stream **150** passes through a queue **129**, which may store up to 5 seconds of traffic. After the Freedom module, ingress traffic stream **150** passes through a WestBay module **128**. This WestBay module places the traffic in a proper form for pooling switch **110**. The form of the traffic is the framing method generic framing protocol (GFP). In another embodiment, the traffic framing method is packet over SONET (POS).

Each physical module has a queue with up to 5 seconds of traffic storage. However, at any given time the amount of traffic stored in the queue is not synchronized between any two physical modules, so if two physical modules receive the same ingress or egress traffic there could be a synchronization difference between the two ingress or egress traffic streams that is as large as the queue. In other words, if the queue has up to 5 seconds of traffic in it, the synchronization error between the two traffic streams could be as large as 5 seconds. This synchronization error is an issue when one of the traffic streams is being used as a back-up traffic stream as part of an automatic protection system (APS) or other system with strict requirements for switching to a backup. In the event of a failure of the traffic stream, the network switching system will switch to the back-up traffic stream. However, because the traffic is not synchronized, there may be either a drop-out or repeat of the traffic equal to the synchronization difference. A network system may not handle these drop-out or repeats well. In some situations, repeats of traffic are worse than drop-outs.

FIG. **2** shows one embodiment of a network switch that does not have a large synchronization difference between two egress traffic streams. In this embodiment, egress traffic stream **242** flows first from service module **200** to pooling switch **210**. Next, egress traffic stream **242** is directed to physical module **230** where it flows through WestBay module **238**, Freedom module **236** (with its queue **239**), TE-MUX module **234**, and link interface module **232** before going out egress link **272**. A second traffic stream is created in link interface module **232** by outputting the same traffic to both the egress link **272** as well as along egress traffic stream **244**. Egress traffic stream **244** travels back through WestBay module **238** and is switched by pooling switch **210** toward physical module **220**. Egress traffic stream **244** flows through WestBay module **228** and then through link interface module **222** to egress link **274**. In this case, all the traffic flows through the same queue **239** in Freedom module **236**. The synchronization difference between the traffic coming out egress link **272** and egress link **274** is then solely due to the time to flow along egress traffic stream **244**. This time can be easily made to be less than the 50 ms requirement of some network specifications. In this embodiment, if the traffic flowing through egress link **272** failed, an APS could switch to the traffic flowing through egress link **274**. In this case, the synchronization difference between the two traffic flows does not have a drop-out or repeat larger than the network can handle. In some embodiments, the synchronization difference is smaller than the specification requirement of less than 50 ms. In some embodiments, the system behaves in a manner to the user as if there is no synchronization difference between the two traffic flows.

FIG. **2** also shows an embodiment of a network switch that does not have a large synchronization difference between two ingress traffic streams. Ingress traffic first enters from ingress link **262** and flows through link interface module **232**, TE-MUX module **234**, Freedom module **236** (with its queue **239**), and WestBay module **238** before leaving physical module **230**. Ingress traffic stream **252** next flows through pooling switch **210** and finally to service module **200**. Another ingress traffic stream enters ingress link **260**. Ingress traffic stream **250** flows through link interface module **222** and WestBay module **228** before leaving physical module **220**. Ingress traffic stream **250** then is switched through pooling switch **210** to physical module **230**. Ingress traffic stream then flows through WestBay module **238** to the link interface **232**. The link interface **232** can choose using switch **290** either the traffic flow from ingress link **260** or ingress link **262** to flow through TE-MUX module **234**, Freedom module **236** (with its queue **239**), and WestBay module **238**. This traffic then ultimately flows through pooling switch **220** to service module **200**. If ingress traffic through ingress link **260** and ingress link **262** are synchronized, then the synchronization difference will be the time for ingress traffic stream **250** to travel from ingress link **260** to link interface **232** in physical module **230**. This synchronization difference can be less than the 50 ms requirement of some network specifications. In this embodiment, if the traffic flowing through ingress link **262** failed, an APS could switch to the traffic flowing through ingress link **260**. In this case, the synchronization difference between these two traffic flows does not have a drop-out or repeat larger than the network can handle. In some embodiments, the synchronization difference is smaller than the specification requirement of less than 50 ms. In some embodiments, the system behaves in a manner to the user as if there is no synchronization difference between the two traffic flows.

In other embodiments of a network switch that does not have a large synchronization difference between two egress traffic streams, pooling switch **210** is not the only pooling switch. A first pooling switch handles traffic only going from a physical module to another physical module. A second pooling switch handles traffic between service module **200** and a physical module. In one embodiment, ingress traffic stream **250** or egress traffic stream **244** are handled by the first pooling switch and ingress traffic stream **252** and egress traffic stream **242** are handled by the second pooling switch.

FIG. **3** shows a reduced cost embodiment of a network switch that does not have a large synchronization difference between two egress traffic streams. In this embodiment, the egress traffic stream **342** flows first from service module **300** to pooling switch **310**. Next, the egress traffic stream is directed to physical module **330** where it flows through WestBay module **338**, Freedom module **336** (with its queue **339**), TE-MUX module **334**, and link interface module **332** before going out egress link **372**. A second traffic stream is created in link interface module **332** by outputting the same traffic to both egress link **372** as well as along egress traffic stream **344**. Egress traffic stream **344** travels back through WestBay module **338** and is switched by pooling switch **310** toward physical module **320**. Egress traffic stream **344** flows through link interface module **322** to egress link **374**. Physical module **320** has fewer components and therefore should have a lower cost. In this case, all the traffic flows through the same queue **339** in Freedom module **336**. The synchronization difference between the traffic coming out egress link **372** and egress link **374** is then solely due to the time to flow along egress traffic stream **344**. This time can be easily made to be less than the 50 ms requirement of some network specifications. In this embodiment, if the traffic flowing through egress link **372** failed, an APS could switch to the traffic flowing through egress link **374**. In this case, the synchronization difference between these two traffic flows will not have a drop-out or repeat larger than the network can handle. Also, the synchronization difference can be smaller than the specification requirement of less than 50 ms.

FIG. **3** also shows a reduced cost embodiment of a network switch that does not have a large synchronization difference between two ingress traffic streams. Ingress traffic first enters from ingress link **362** and flows through link interface module **332**, TE-MUX module **334**, Freedom module **336** (with its queue **339**), and WestBay module **338** before leaving physical module **330**. The ingress traffic stream **352** next flows through pooling switch **310** and finally to service module **300**. Another ingress traffic stream enters ingress link **360**. Ingress traffic stream **350** flows through link interface module **322** and then leaves physical module **320**. Physical module **320** has fewer components and therefore should have a lower cost. Ingress traffic stream **350** then is switched through pooling switch **310** to physical module **330**. Ingress traffic stream then flows through WestBay module **338** to link interface **332**. Link interface **332** can choose using switch **390** either the traffic flow from ingress link **360** or ingress link **362** to flow through TE-MUX module **334**, Freedom module **336** (with its queue **339**), and WestBay module **338**. This traffic then ultimately flows through pooling switch **320** to service module **300**. If ingress traffic through ingress link **360** and ingress link **362** are synchronized, then the synchronization difference will be the time for ingress traffic stream **350** to travel from ingress link **360** to link interface **332** in physical module **330**. This synchronization difference can be easily made to be less than the 50 ms requirement of some network specifications. In this embodiment, if the traffic flowing through ingress link **362** failed, an APS could switch to the traffic flowing through ingress link **360**. In this case, the synchronization difference between these two traffic flows will not have a drop-out or repeat larger than the network can handle. Also, the synchronization difference can be smaller than the specification requirement of less than 50 ms.

In another embodiment of a reduced cost network switch that does not have a large synchronization difference between two egress traffic streams, the pooling switch **310** is not only one pooling switch. A first pooling switch handles traffic only going from a physical module to another physical module. A second pooling switch handles traffic between service module **300** and a physical module. For example, ingress traffic stream **350** or egress traffic stream **344** is handled by the first pooling switch and ingress traffic stream **352** and egress traffic stream **342** is handled by the second pooling switch.

FIG. **4** shows a further cost reduced embodiment of a network switch that does not have a large synchronization difference between multiple egress traffic streams. In this embodiment, egress traffic stream **442** flows first from service module **400** to pooling switch **410**. Next, egress traffic stream **442** is directed to physical module **430** where it flows through WestBay module **438**, Freedom module **436** (with its queue **439**), TE-MUX module **434**, and link interface module **432** before going out egress link **472**. A second traffic stream is created in link interface module **432** by outputting the same traffic to both egress link **472** as well as along egress traffic stream **444**. Egress traffic stream **444** travels back through WestBay module **438** and is switched by pooling switch **410** toward physical module **480**. The egress traffic stream **444** flows through link interface module **482** to one of the egress links **474**. Notice that physical module **480** has fewer components and therefore should have a lower cost. Additionally, physical module **491** producing egress traffic stream **445** could use one of the egress links **474** as an alternate or back-up. Similarly, physical module **492** producing an egress traffic stream (not shown in FIG. **4**) could also use one of egress links **474** (also not shown in FIG. **4**) as an alternate or back-up. In a similar way, multiple egress traffic streams could use a physical module as an alternate or back-up.

In another embodiment, pooling switch **410** is not only one pooling switch. A first pooling switch could handle traffic only going from a physical module to another physical module. A second pooling switch could handle traffic between the service module **400** and a physical module.

FIG. **5** shows another embodiment of a network switch. In this embodiment, egress/ingress traffic stream **542** going to and from service module **500**, through physical module **530** to egress/ingress link **572** has an alternate or back-up through egress/ingress traffic stream **544** and one of the egress/ingress links **574**. Similarly, egress/ingress traffic stream **543** going to and from service module **500**, through physical module **591** to egress/ingress link (not shown in FIG. **5**) has an alternate or back-up through egress/ingress traffic stream **545** and one of the egress/ingress links **574**. To provide an APS for the physical modules (for example, physical modules **530** or **591**), pooling switch **510** can switch to physical module **592**. This is an example of 1:1 or 1+1 redundancy for the physical modules like **592** and **591**. In other words, one physical module (physical module **592**) is a back-up for another physical module (in this physical module **591**). So, if for example, physical module **530** failed, pooling switch **510** could use back-up egress/ingress links **574** and direct the traffic along egress/ingress traffic stream **546** to physical module **592** and then again through the pooling switch along egress/ingress traffic stream **547** to service module **500**. In some embodiments, the pooling switch **510** directs parts of the egress/ingress traffic stream **545** to physical modules **591** and **592** for reasons other than backup. In some embodiments, parts of the traffic stream are directed to physical modules **591** and **592** to better use the bandwidth capability of the physical modules **591** and **592**. In some embodiments, parts of the traffic stream are directed to physical modules **591** and **592** because parts of the traffic stream require different protocol processing and physical modules **591** and **592** have different protocol processing capabilities.

FIG. **6** is a block diagram illustrating another embodiment of an APS using a pooling switch. Pooling switch **612** switches traffic between physical modules and is a part of an APS. In this embodiment, egress/ingress traffic stream **642** starts/ends in service module **600** travels through pooling switch **610** to physical module **630**. This traffic is processed through the physical module and then instead of being output to a egress/ingress link, the traffic is sent/received to a second pooling switch **612**. The physical module **630** has no link interface module such as an optical or electrical interface. The second pooling switch **612** directs the traffic stream to/from both physical module **682** and physical module **680**. The egress/ingress traffic stream **646** travels through physical module **682** and finally through egress/ingress links **672**. The egress/ingress traffic stream **644** travels through physical module **680** and finally through egress/ingress links **674**. Similarly, egress/ingress traffic stream **643** starts/ends in service module **600** travels through pooling switch **610** to physical module **691**. This traffic is processed through the physical module and then instead of being output to a egress/ingress link from the link interface in physical module **691**, the traffic is sent/received to a second pooling switch **612**. The second pooling switch **612** directs the traffic stream to/from both physical module **682** and physical module **680**. Egress/ingress traffic stream **647** travels through physical module **682** and finally through egress/ingress links **672**. Egress/ingress traffic stream **645** travels through physical module **680** and finally through egress/ingress links **674**. In another embodiment, the two pooling switches **610** and **612** are combined into one physical switch that can perform both functions.

In some embodiments, pooling switch **612** helps to utilize physical module resources efficiently. For example, egress/ingress traffic stream **644** could be directed, on a granularity of a smaller logical unit (e.g. STS-1 frame), by pooling switch **612** to different physical modules (e.g. physical module **630**, **691**, and **692**). Different physical modules (e.g. **630**, **691**, and **692**) can be used to process parts of the traffic stream coming across one egress/ingress link.

In some embodiments, pooling switch **612** helps to utilize physical module resources based on different protocols or different densities. Different physical modules may process different protocols more efficiently than others, and the pooling switch can be used to direct the protocols accordingly. Different physical modules may have different bandwidth capabilities, and the pooling switch can be used to direct traffic according to what might be the most efficient use of the hardware in the modules.

In some embodiments, pooling switch **612** can also be used to provide 1:N redundancy for traffic streams where the traffic, upon a failure, would be directed to the physical module devoted to providing back-up.

Although the foregoing embodiments have been described in some detail for purposes of clarity of understanding, the invention is not limited to the details provided. There are many alternative ways of implementing the invention. The disclosed embodiments are illustrative and not restrictive.

What is claimed is:

1. A method of protecting a protected egress link including:

connecting traffic from a service module to a first physical module having a link layer framer that is connected via a link interface of the first physical module to the protected egress link, wherein the link layer framer includes a queue for storing the traffic; and

connecting the traffic input to the link interface of the first physical module through a pooling switch to a second physical module that is connected via a link interface of the second physical module to an alternate egress link, wherein the traffic in the second physical module is not processed through a link layer framer of the second physical module but is processed through the link layer framer of the first physical module.

2. A method of protecting a protected egress link as in claim **1**, wherein the link interface of the first physical module comprises an optical link interface module.

3. A method of protecting a protected egress link as in claim **1**, wherein the link interface of the second physical module comprises an optical link interface module.

4. A method of protecting a protected egress link as in claim **1**, wherein the link interface of the first physical module comprises an electrical link interface module.

5. A method of protecting a protected egress link as in claim **1**, wherein the link interface of the second physical module comprises an electrical link interface module.

6. A method of protecting a protected egress link as in claim **1**, wherein the first physical module comprises a module that places the traffic in proper form for a pooling switch.

7. A method of protecting a protected egress link as in claim **1**, wherein the second physical module comprises a module that places the traffic in proper form for a pooling switch.

8. A method of protecting a protected egress link as in claim **1**, wherein the traffic through the protected egress link and the alternate egress link have a synchronization difference smaller than 50 ms.

9. A method of protecting a protected egress link as in claim **1**, wherein the traffic through the protected egress link and the alternate egress link behave in a manner to the user as if there is no synchronization difference between the two traffic flows.

10. A method of protecting a protected egress link as in claim **1**, wherein the pooling switch enables multiple logical streams to be included in one physical interface.

11. A method of protecting a protected egress link as in claim **1**, wherein the pooling switch is a packet switch.

12. A method of protecting a protected egress link as in claim **1**, wherein pooling switch is a time division multiplexing switch.

13. A method of protecting a protected ingress link including:

connecting traffic to a service module from a first physical module having a link layer framer that is connected via a link interface of the first physical module to the protected ingress link, wherein the link layer framer includes a queue for storing the traffic; and

connecting the traffic through a second physical module that is connected via a link interface of the second physical module to an alternate ingress link through a pooling switch to the link interface of the first physical module, wherein the traffic in the second physical module is not processed through a link layer framer of the second physical module but is processed through the link layer framer of the first physical module in the event that the traffic through the second physical module from the alternate ingress link is selected to be used.

14. A method of protecting a protected egress link as in claim **13**, wherein the service module decides from information within an input traffic stream to the service module where to output the input traffic stream.

15. A method of protecting a protected ingress link as in claim **13**, wherein the link interface of the first physical module comprises an optical link interface module.

16. A method of protecting a protected ingress link as in claim **13**, wherein the link interface of the second physical module comprises an optical link interface module.

17. A method of protecting a protected ingress link as in claim **13**, wherein the link interface of the first physical module comprises an electrical link interface module.

18. A method of protecting a protected ingress link as in claim **13**, wherein the link interface of the second physical module comprises an electrical link interface module.

19. A method of protecting a protected ingress link as in claim **13**, wherein the first physical module contains a module that places the traffic in proper form for a pooling switch.

20. A method of protecting a protected ingress link as in claim **13**, wherein the second physical module contains a module that places the traffic in proper form for a pooling switch.

21. A method of protecting a protected ingress link as in claim **13**, wherein the traffic through the protected ingress link and the alternate ingress link have a synchronization difference smaller than 50 ms.

22. A method of protecting a protected ingress link as in claim **13**, wherein the traffic through the protected ingress link and the alternate ingress link behave in a manner to the user as if there is no synchronization difference between the two traffic flows.

23. A method of protecting a protected ingress link as in claim **13**, wherein the pooling switch enables multiple logical streams to be included in one physical interface.

24. A method of protecting a protected ingress link as in claim **13**, wherein the pooling switch is a packet switch.

25. A method of protecting a protected ingress link as in claim **13**, wherein pooling switch is a time division multiplexing switch.

26. A method of protecting a protected egress link including:

connecting traffic from a service module to a first pooling switch;

connecting the first pooling switch to a first physical module having a link layer framer that is connected via a link interface of the first physical module to the protected egress link, wherein the link layer framer includes a queue for storing the traffic; and

connecting the traffic input to the link interface of the first physical module through a second pooling switch to a second physical module that is connected via a link interface of the second physical module to an alternate egress link, wherein the traffic in the second physical module is not processed through a link layer framer of the second physical module but is processed through the link layer framer of the first physical module.

27. A method of protecting a protected ingress link including:

connecting traffic to a service module from a first pooling switch;

connecting the first pooling switch to a first physical module having a link layer framer that is connected via a link interface of the first physical module to the protected ingress link, wherein the link layer framer includes a queue for storing the traffic; and

connecting the traffic through a second physical module that is connected via a link interface of the second physical module to an alternate ingress link through a second pooling switch to the link interface of the first physical module, wherein the traffic in the second physical module is not processed through a link layer framer of the second physical module but is processed through the link layer framer of the first physical module in the event that the traffic through the second physical module from the alternate ingress link is selected to be used.

28. A method of protecting a protected egress link including:

connecting traffic from a service module to a first pooling switch;

connecting the first pooling switch to a first physical module having a link layer framer, wherein the link layer framer includes a queue for storing the traffic;

connecting the traffic through the first physical module through a second pooling switch to a second physical module that is connected via a link interface of the second physical module to a protected egress link, wherein the traffic in the second physical module is not processed through a link layer framer of the second physical module but is processed by the link layer framer of the first physical module; and

connecting the traffic through the first physical module through the second pooling switch to a third physical module that is connected via a link interface of the third physical module to an alternate egress link, wherein the traffic in the third physical module is not processed through a link layer framer of the third physical module but is processed by the link layer framer of the first physical module.

29. A method of protecting a protected egress link as in claim **28**, wherein the first physical module does not include a link interface module.

30. A method of protecting a protected egress link as in claim **28**, wherein 1:N protection is provided.

31. A method of protecting a protected ingress link including:

connecting traffic to a service module from a first pooling switch;

connecting the first pooling switch to a first physical module having a link layer framer, wherein the link layer framer includes a queue for storing the traffic;

connecting the traffic through a second physical module that is connected via a link interface of the second physical module to a protected ingress link through a second pooling switch to the first physical module, wherein the traffic in the second physical module is not processed through a link layer framer of the second physical module but is processed by the link layer framer of the first physical module; and

connecting the traffic through a third physical module that is connected via a link interface of the third physical module to an alternate ingress link through the second pooling switch to the first physical module, wherein the traffic in the third physical module is not processed through a link layer framer of the third physical module but is processed by the link layer framer of the first physical module.

32. A method of protecting a protected ingress link as in claim **31**, wherein the first physical module does not include a link interface module.

33. A method of protecting a protected ingress link as in claim **31**, wherein 1:N protection is provided.

34. A system for protecting a protected egress link including:

a service module;

a first physical module, wherein the first physical module comprises a link layer framer that is connected via a link interface to the protected egress link, and wherein the link layer framer includes a queue for storing traffic, and wherein traffic is connected from the service module to the first physical module;

a pooling switch; and

a second physical module, wherein the second physical module is connected via a link interface of the second physical module to an alternate egress link, and wherein traffic in the second physical module is not processed through a link layer framer of the second physical module but is processed through a link layer framer of the first physical module, and wherein traffic input to the link interface of the first physical module through the polling switch to the second physical module.

35. A system for protecting a protected egress link as in claim **34**, wherein the link interface of the first physical module comprises an optical link interface module.

36. A system for protecting a protected egress link as in claim **34**, wherein the link interface of the second physical module comprises an optical link interface module.

37. A system for protecting a protected egress link as in claim **34**, wherein the link interface of the first physical module comprises an electrical link interface module.

38. A system for protecting a protected egress link as in claim **34**, wherein the link interface of the second physical module comprises an electrical link interface module.

39. A system for protecting a protected egress link as in claim **34**, wherein the first physical module comprises a module that places the traffic in proper form for a pooling switch.

40. A system for protecting a protected egress link as in claim **34**, wherein the second physical module comprises a module that places the traffic in proper form for a pooling switch.

41. A system for protecting a protected egress link as in claim **34**, wherein the traffic through the protected egress link and the alternate egress link have a synchronization difference smaller than 50 ms.

42. A system for protecting a protected egress link as in claim **34**, wherein the traffic through the protected egress link and the alternate egress link behave in a manner to the user as if there is no synchronization difference between the two traffic flows.

43. A system for protecting a protected egress link as in claim **34**, wherein the pooling switch enables multiple logical streams to be included in one physical interface.

44. A system for protecting a protected egress link as in claim **34**, wherein the pooling switch is a packet switch.

45. A system for protecting a protected egress link as in claim **34**, wherein pooling switch is a time division multiplexing switch.

46. A system for protecting a protected ingress link including:

- a service module;
- a first physical module, wherein the first physical module comprises a link layer framer that is connected via a link interface to the protected ingress link, and wherein the link layer framer includes a queue for storing traffic, and wherein traffic is connected from the service module to the first physical module;
- a pooling switch; and
- a second physical module, wherein the second physical module is connected via a link interface of the second physical module to an alternate ingress link, and wherein traffic in the second physical module is not processed through a link layer framer of the second physical module but is processed through a link layer framer of the first physical module in the event that the traffic through the second physical module from the alternate ingress link is selected to be used, and wherein traffic is connected through the second physical module.

47. A system for protecting a protected egress link as in claim **46**, wherein the service module decides from information within an input traffic stream to the service module where to output the input traffic stream.

48. A system for protecting a protected egress link as in claim **46**, wherein the link interface of the first physical module comprises an optical link interface module.

49. A system for protecting a protected egress link as in claim **46**, wherein the link interface of the second physical module comprises an optical link interface module.

50. A system for protecting a protected egress link as in claim **46**, wherein the link interface of the first physical module comprises an electrical link interface module.

51. A system for protecting a protected egress link as in claim **46**, wherein the link interface of the second physical module comprises an electrical link interface module.

52. A system for protecting a protected egress link as in claim **46**, wherein the first physical module contains a module that places the traffic in proper form for a pooling switch.

53. A system for protecting a protected egress link as in claim **46**, wherein the second physical module contains a module that places the traffic in proper form for a pooling switch.

54. A system for protecting a protected egress link as in claim **46**, wherein the traffic through the protected ingress link and the alternate ingress link have a synchronization difference smaller than 50 ms.

55. A system for protecting a protected egress link as in claim **46**, wherein the traffic through the protected ingress link and the alternate ingress link behave in a manner to the user as if there is no synchronization difference between the two traffic flows.

56. A system for protecting a protected egress link as in claim **46**, wherein the pooling switch enables multiple logical streams to be included in one physical interface.

57. A system for protecting a protected egress link as in claim **46**, wherein the pooling switch is a packet switch.

58. A system for protecting a protected egress link as in claim **46**, wherein pooling switch is a time division multiplexing switch.

59. A system for protecting a protected egress link including:

- a service module;
- a first pooling switch, wherein traffic is connected from the service module to the first pooling switch;
- a first physical module, wherein the first pooling switch is connected to the first physical module having a link layer framer that is connected via a link interface of the first physical module to the protected egress link, wherein the link layer framer includes a queue for storing the traffic; and
- a second pooling switch;
- a second physical module, wherein the traffic input to the link interface of the first physical module is connected through the second pooling switch to the second physical module that is connected via a link interface of the second physical module to an alternate egress link, wherein the traffic in the second physical module is not processed through a link layer framer of the second physical module but is processed through the link layer framer of the first physical module.

60. A system for protecting a protected ingress link including:

- a service module;
- a first pooling switch, wherein traffic is connected to the service module from the first pooling switch;
- a first physical module, wherein the first pooling switch is connected to a first physical module having a link layer framer that is connected via a link interface of the first physical module to the protected ingress link, wherein the link layer framer includes a queue for storing the traffic; and
- a second physical module, wherein the traffic is connected through the second physical module that is connected via a link interface of the second physical module to an alternate ingress link through a second pooling switch to the link interface of the first physical module, wherein

the traffic in the second physical module is not processed through a link layer framer of the second physical module but is processed through the link layer framer of the first physical module in the event that the traffic through the second physical module from the alternate ingress link is selected to be used.

61. A system for protecting a protected egress link including:

a service module;

a first pooling switch, wherein traffic is connected from a service module to a first pooling switch;

a first physical module, wherein the first pooling switch is connected to the first physical module having a link layer framer, wherein the link layer framer includes a queue for storing the traffic;

a second pooling switch, wherein the traffic is connected through the first physical module through the second pooling switch to a second physical module that is connected via a link interface of the second physical module to a protected egress link, wherein the traffic in the second physical module is not processed through a link layer framer of the second physical module but is processed by the link layer framer of the first physical module; and

a third physical module, wherein connecting the traffic is connected through the first physical module through the second pooling switch to the third physical module that is connected via a link interface of the third physical module to an alternate egress link, wherein the traffic in the third physical module is not processed through a link layer framer of the third physical module but is processed by the link layer framer of the first physical module.

62. A system for protecting a protected egress link as in claim **61**, wherein the first physical module does not include a link interface module.

63. A system for protecting a protected egress link as in claim **61**, wherein 1:N protection is provided.

64. A system for protecting a protected ingress link including:

a service module;

a first pooling switch, wherein traffic is connected to the service module from the first pooling switch;

a first physical module, wherein the first pooling switch is connected to the first physical module having a link layer framer, wherein the link layer framer includes a queue for storing the traffic;

the second physical module, wherein the traffic is connected through the second physical module that is connected via a link interface of the second physical module to a protected ingress link through a second pooling switch to the first physical module, wherein the traffic in the second physical module is not processed through a link layer framer of the second physical module but is processed by the link layer framer of the first physical module; and

a third physical module, wherein the traffic is connected through the third physical module that is connected via a link interface of the third physical module to an alternate ingress link through the second pooling switch to the first physical module, wherein the traffic in the third physical module is not processed through a link layer framer of the third physical module but is processed by the link layer framer of the first physical module.

65. A system for protecting a protected ingress link as in claim **64**, wherein the first physical module does not include a link interface module.

66. A system for protecting a protected ingress link as in claim **64**, wherein 1:N protection is provided.

* * * * *

Exhibit C

US007899916B1

(12) **United States Patent**
Bumstead et al.

(10) **Patent No.: US 7,899,916 B1**
(45) **Date of Patent: Mar. 1, 2011**

(54) **VIRTUAL SERVICE ENDPOINT**

(75) Inventors: **David Bumstead**, San Jose, CA (US); **John Burns**, Los Altos, CA (US); **Fong Liaw**, San Jose, CA (US); **Jan Medved**, Pleasanton, CA (US); **John Z. Yu**, Santa Clara, CA (US)

(73) Assignee: **Brixham Solutions Ltd.**, Tortola (VG)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1032 days.

(21) Appl. No.: **10/677,090**

(22) Filed: **Sep. 30, 2003**

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/228**; 709/227; 709/231

(58) **Field of Classification Search** 709/227, 709/228, 231
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,610,918	A	3/1997	Kamo et al.	
5,675,642	A	10/1997	Sone	
5,764,750	A *	6/1998	Chau et al.	379/229
5,905,873	A	5/1999	Hartmann et al.	
6,047,002	A	4/2000	Hartmann et al.	
6,201,809	B1	3/2001	Lewin et al.	
6,205,152	B1	3/2001	Von Ahnen et al.	
7,065,585	B2	6/2006	Ohyoshi et al.	
7,082,121	B1 *	7/2006	Stammers et al.	370/352
7,590,143	B2 *	9/2009	Daley et al.	370/466
2002/0065073	A1	5/2002	Natani et al.	
2003/0081624	A1	5/2003	Aggarwal et al.	
2003/0126195	A1 *	7/2003	Reynolds et al.	709/203
2004/0120502	A1 *	6/2004	Strathmeyer et al.	379/265.01

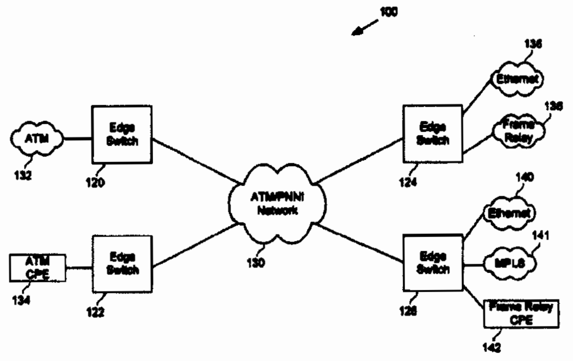
* cited by examiner

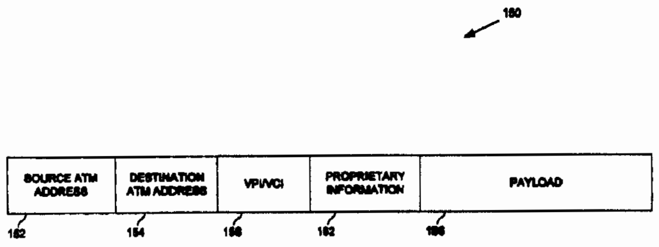
Primary Examiner—Barbara N Burgess

(57) **ABSTRACT**

A system and method are disclosed for establishing a connection between a first endpoint participating in a first protocol and a second endpoint participating in a second protocol. Establishing a connection includes establishing a virtual service endpoint participating in the first protocol on a network device that is connected to the first endpoint and the second endpoint and forwarding to the second endpoint communication directed from the first endpoint to the virtual service endpoint.

21 Claims, 12 Drawing Sheets





ATM Packet Format Carrying Proprietary Information

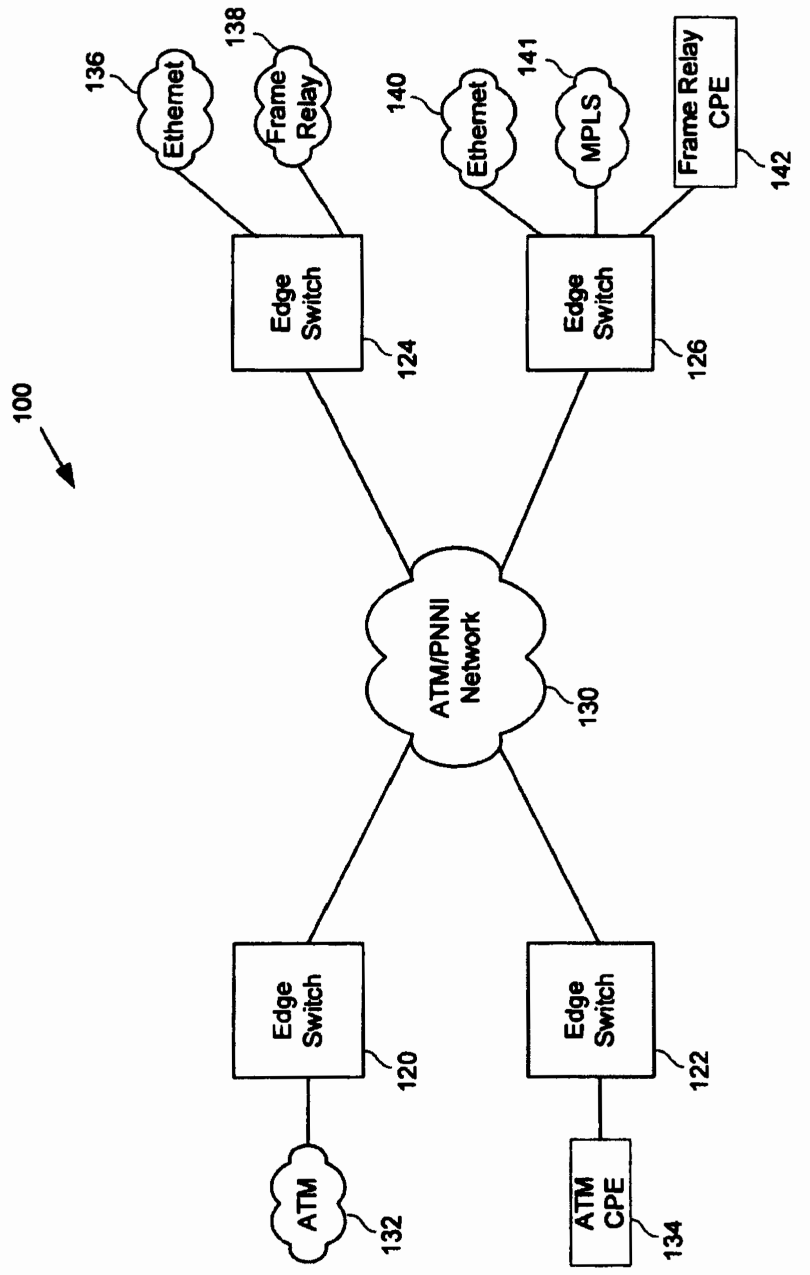

100
Ethernet
136
Frame Relay
138
Ethernet
140
MPLS
141
Frame Relay CPE
142
Edge Switch
124
Edge Switch
126
ATM/PNNI Network
130
Edge Switch
120
Edge Switch
122
ATM
132
ATM CPE
134

FIG 1A

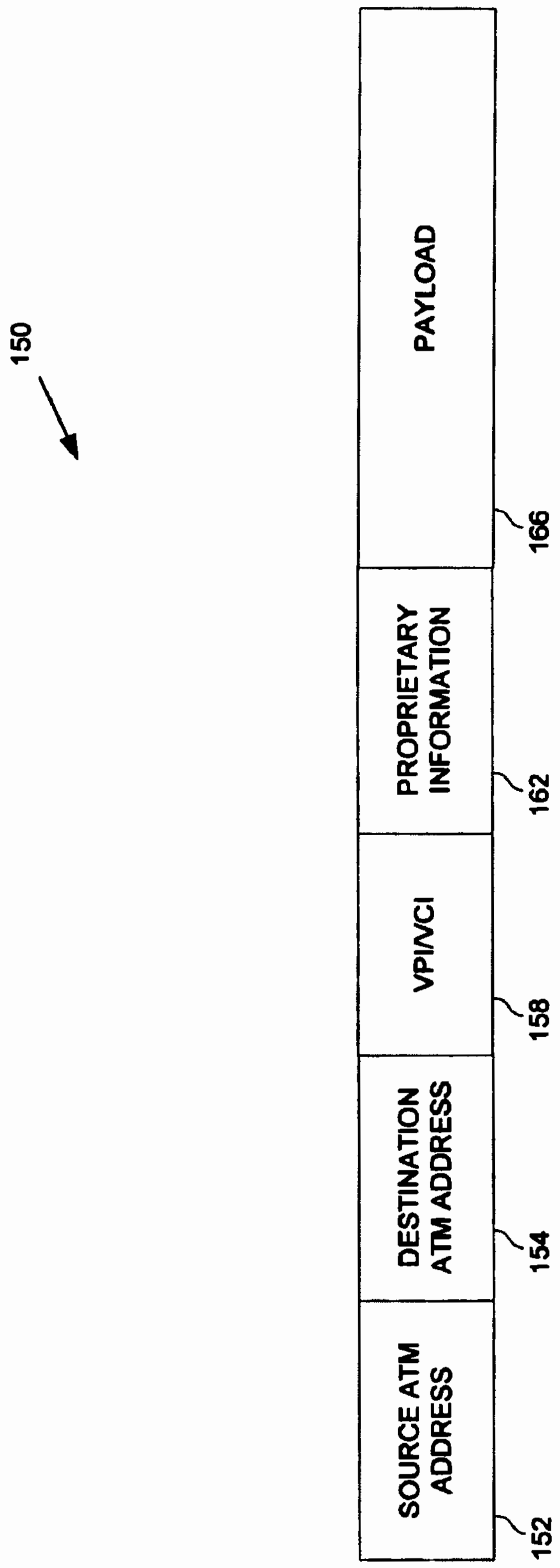


FIG 1B - ATM Packet Format Carrying Proprietary Information

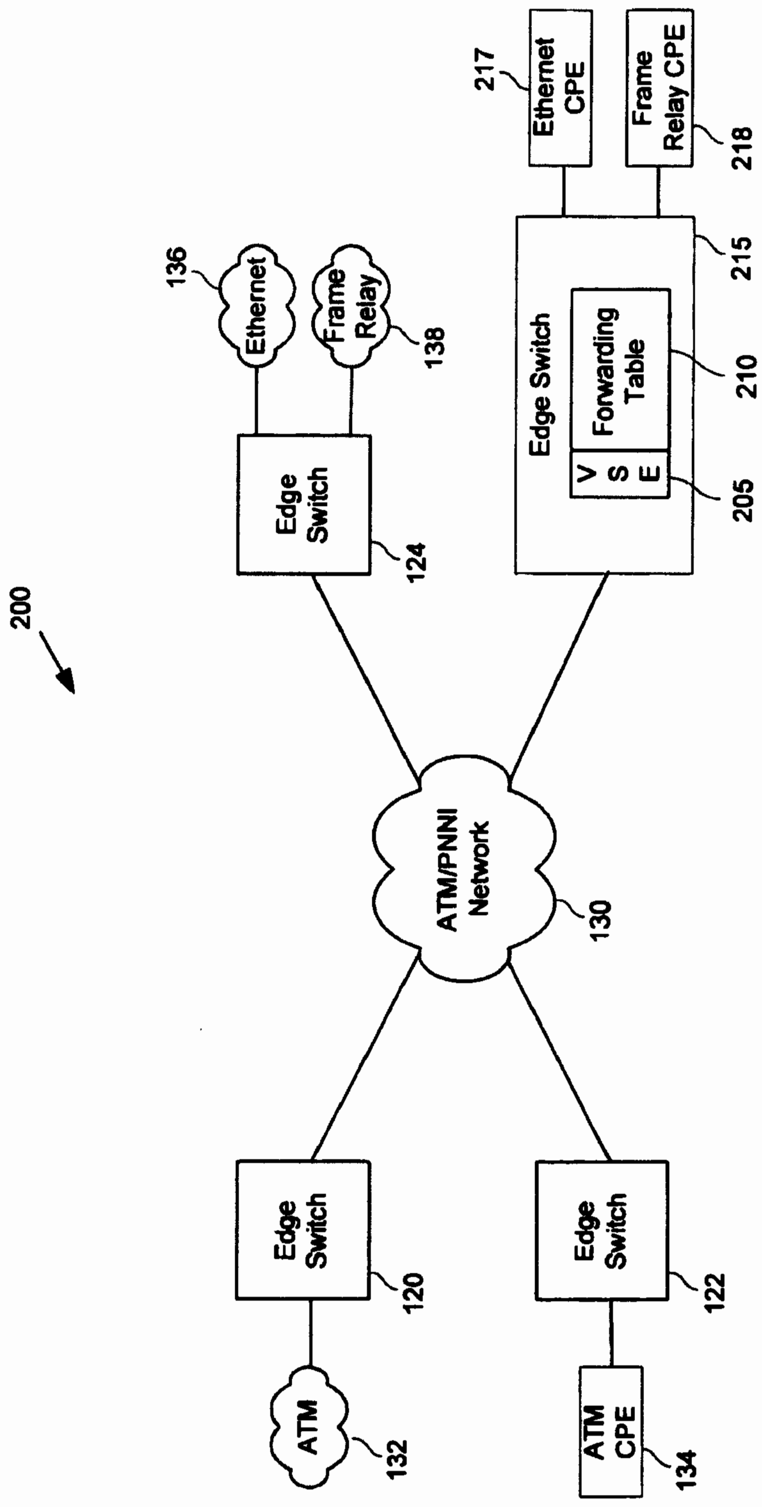


FIG 2A - Edge Switch with VSE

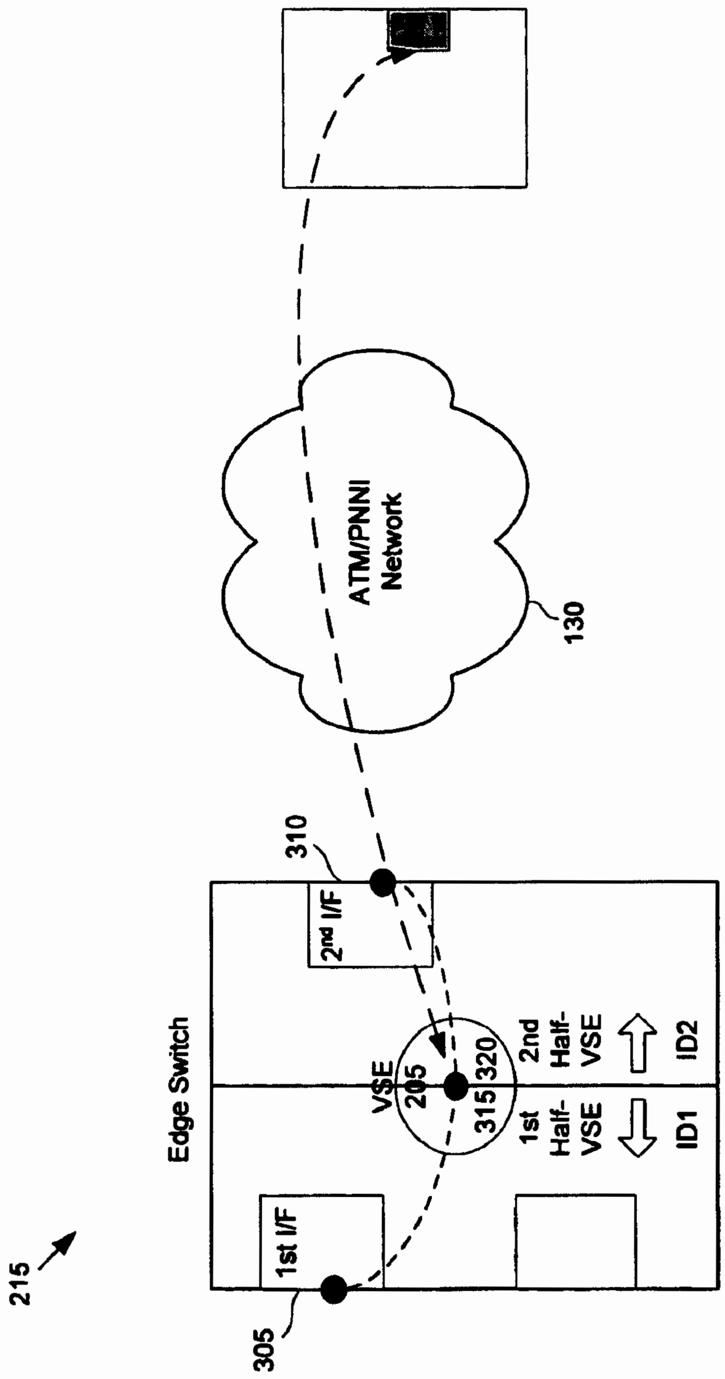


FIG 3A - VSE for Edge Service Interworking Application

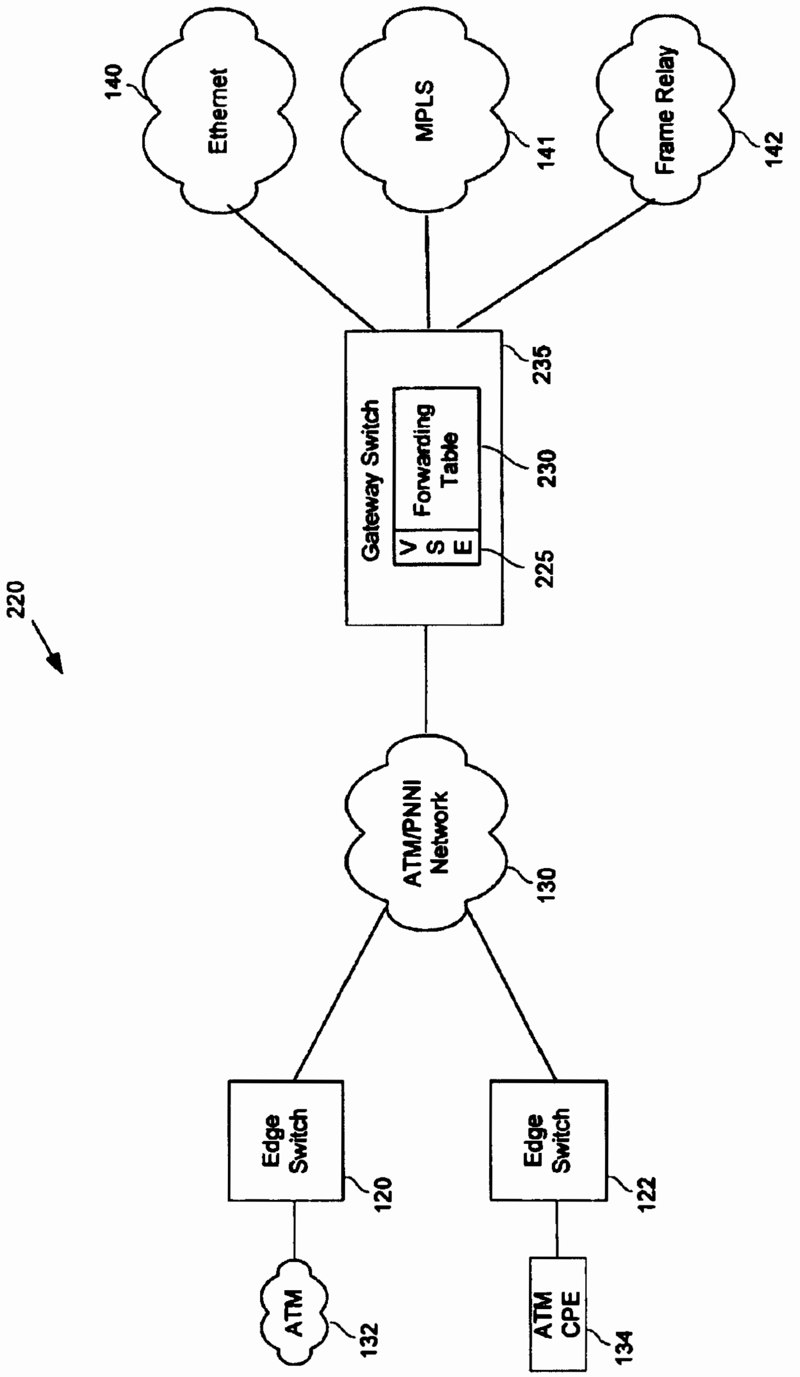
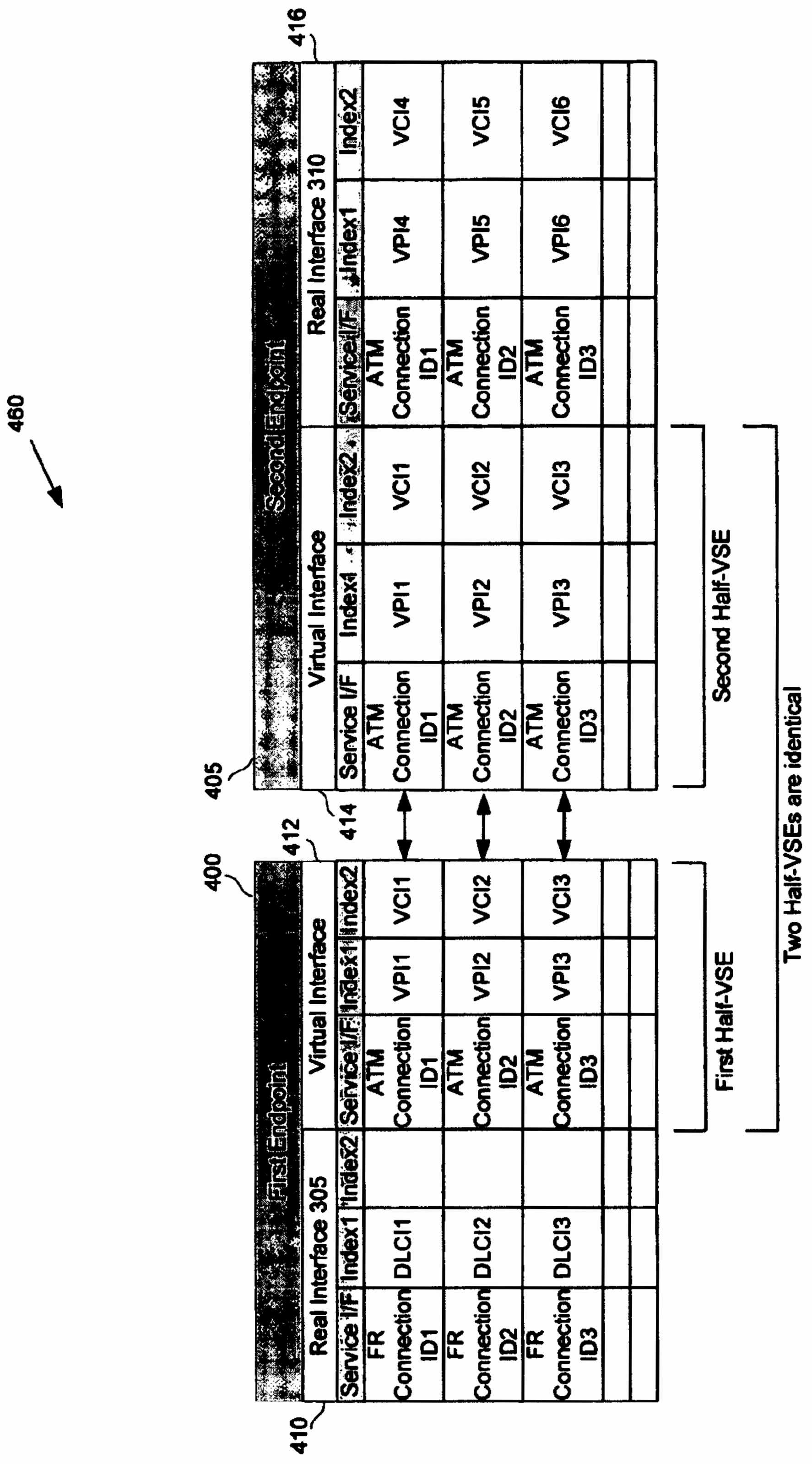


FIG 2B - Gateway Switch with VSE



First Endpoint

Real Interface 305			Virtual Interface		
Service I/F	Index1	Index2	Service I/F	Index1	Index2
FR Connection ID1	DLCI1		ATM Connection ID1	VPI1	VCI1
FR Connection ID2	DLCI2		ATM Connection ID2	VPI2	VCI2
FR Connection ID3	DLCI3		ATM Connection ID3	VPI3	VCI3

Second Endpoint

Virtual Interface			Real Interface 310		
Service I/F	Index1	Index2	Service I/F	Index1	Index2
ATM Connection ID1	VPI1	VCI1	ATM Connection ID1	VPI4	VCI4
ATM Connection ID2	VPI2	VCI2	ATM Connection ID2	VPI5	VCI5
ATM Connection ID3	VPI3	VCI3	ATM Connection ID3	VPI6	VCI6

FIG 3B - Forwarding Tables for Edge Service Interworking

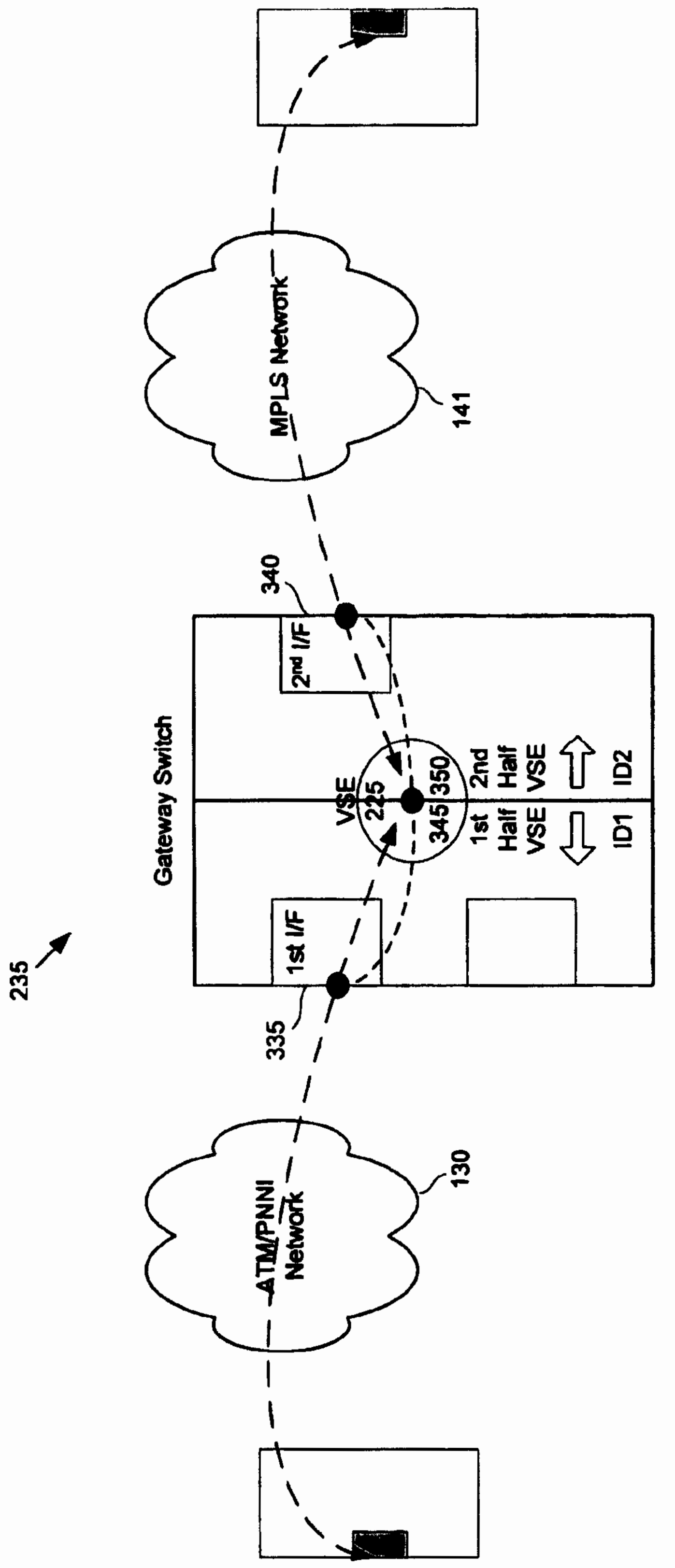
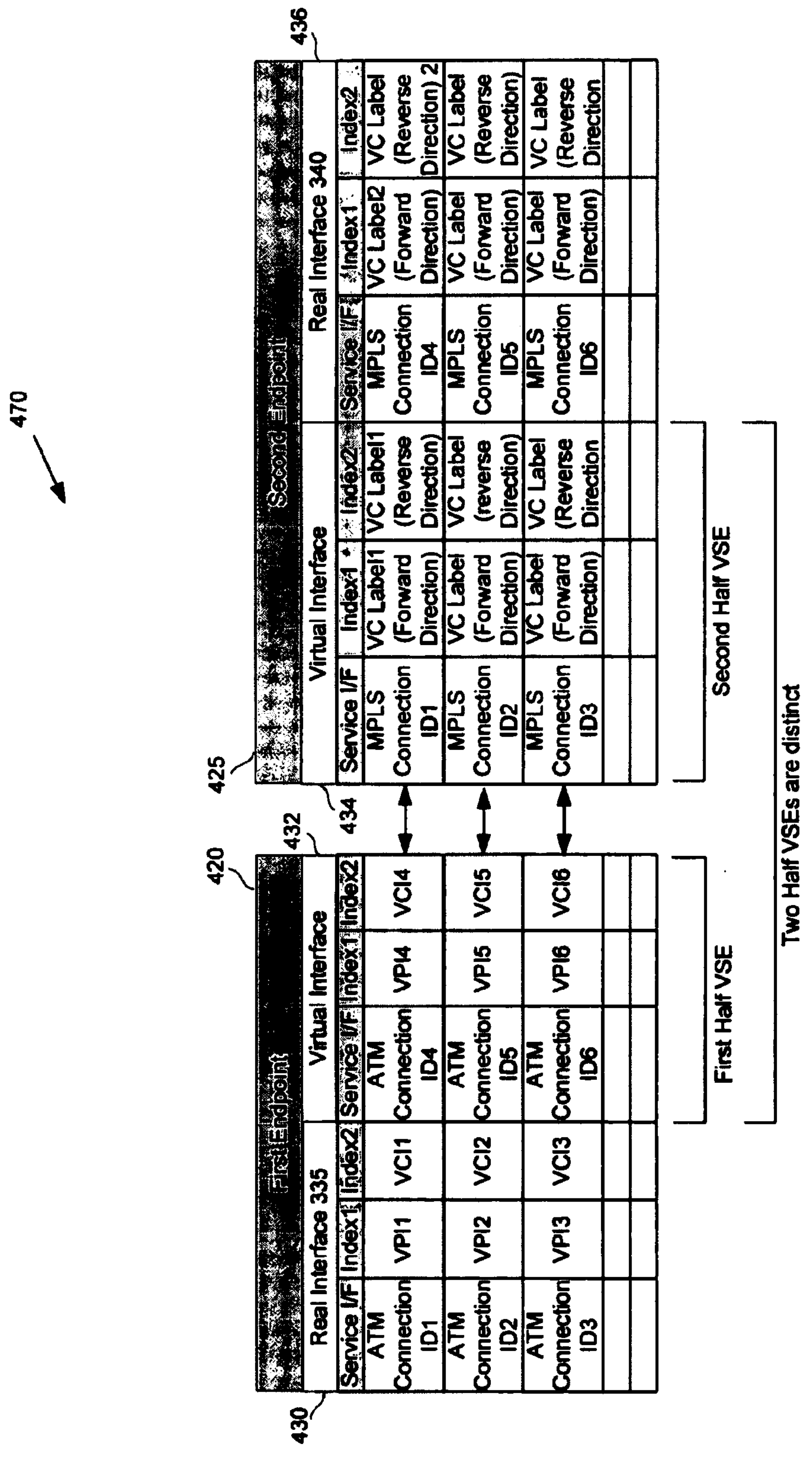


FIG 4A - VSE for Gateway Network Interworking Application



First Endpoint

Real Interface 335			Virtual Interface		
Service I/F	Index1	Index2	Service I/F	Index1	Index2
ATM Connection ID1	VPI1	VCI1	ATM Connection ID4	VPI4	VCI4
ATM Connection ID2	VPI2	VCI2	ATM Connection ID5	VPI5	VCI5
ATM Connection ID3	VPI3	VCI3	ATM Connection ID6	VPI6	VCI6

Second Endpoint

Virtual Interface			Real Interface 340		
Service I/F	Index1	Index2	Service I/F	Index1	Index2
MPLS Connection ID1	VC Label1 (Forward Direction)	VC Label1 (Reverse Direction)	MPLS Connection ID4	VC Label2 (Forward Direction)	VC Label (Reverse Direction) 2
MPLS Connection ID2	VC Label (Forward Direction)	VC Label (reverse Direction)	MPLS Connection ID5	VC Label (Forward Direction)	VC Label (Reverse Direction)
MPLS Connection ID3	VC Label (Forward Direction)	VC Label (Reverse Direction	MPLS Connection ID6	VC Label (Forward Direction)	VC Label (Reverse Direction

FIG 4B - Forwarding Tables for Gateway Inter-networking

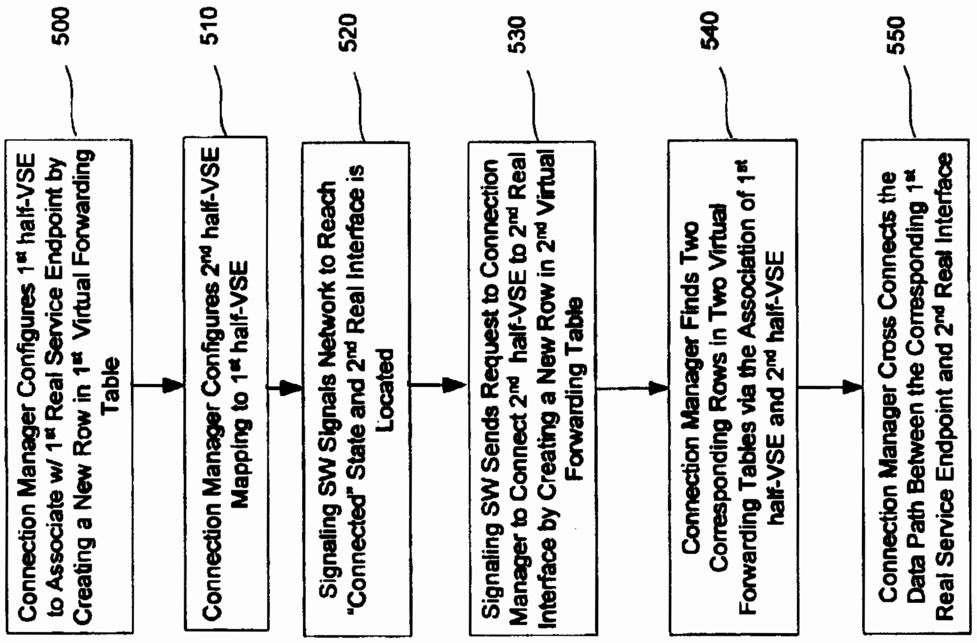

500
Connection Manager Configures 1st half-VSE to Associate w/ 1st Real Service Endpoint by Creating a New Row in 1st Virtual Forwarding Table
510
Connection Manager Configures 2nd half-VSE Mapping to 1st half-VSE
520
Signaling SW Signals Network to Reach "Connected" State and 2nd Real Interface is Located
530
Signaling SW Sends Request to Connection Manager to Connect 2nd half-VSE to 2nd Real Interface by Creating a New Row in 2nd Virtual Forwarding Table
540
Connection Manager Finds Two Corresponding Rows in Two Virtual Forwarding Tables via the Association of 1st half-VSE and 2nd half-VSE
550
Connection Manager Cross Connects the Data Path Between the Corresponding 1st Real Service Endpoint and 2nd Real Interface

FIG 5 - Establishing a Connection - Edge Service Interworking Application

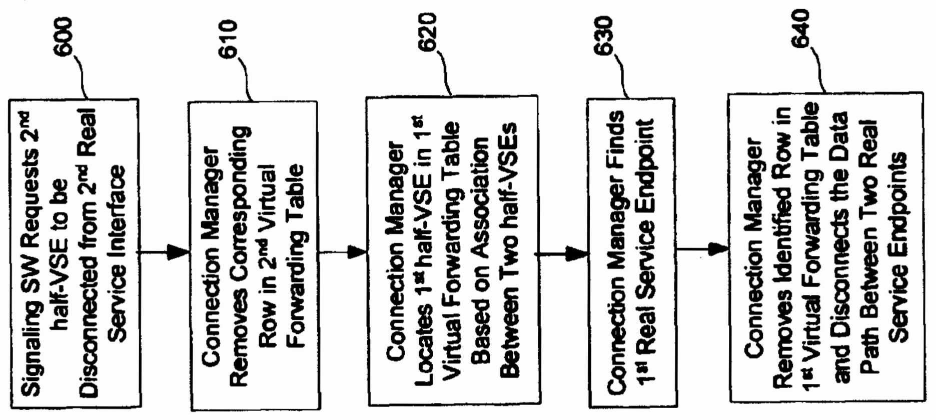

600
Signaling SW Requests 2nd half-VSE to be Disconnected from 2nd Real Service Interface
610
Connection Manager Removes Corresponding Row in 2nd Virtual Forwarding Table
620
Connection Manager Locates 1st half-VSE in 1st Virtual Forwarding Table Based on Association Between Two half-VSEs
630
Connection Manager Finds 1st Real Service Endpoint
640
Connection Manager Removes Identified Row in 1st Virtual Forwarding Table and Disconnects the Data Path Between Two Real Service Endpoints

FIG 6 - Releasing a Connection - Edge Service Interworking Application

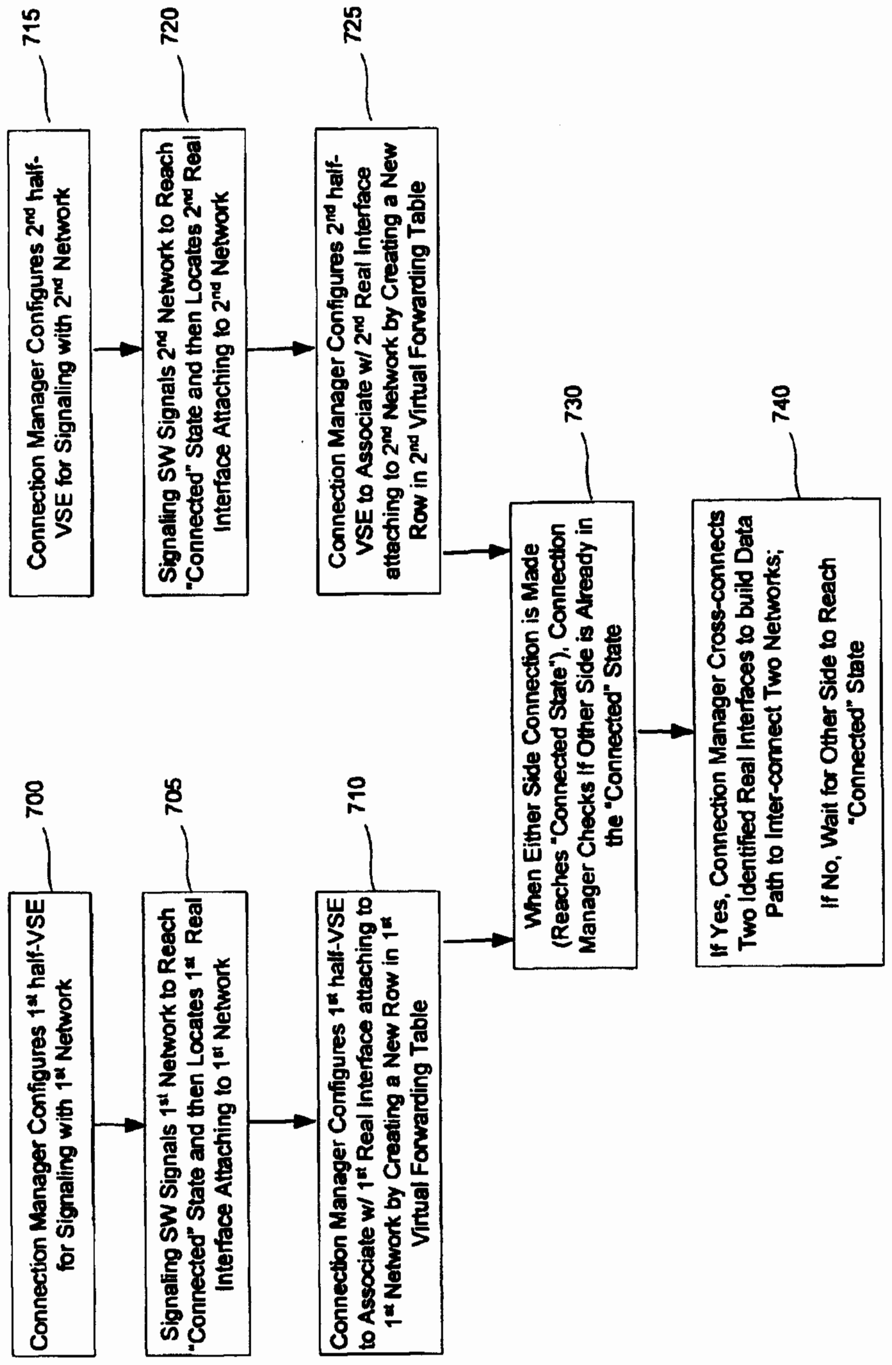

700
Connection Manager Configures 1st half-VSE for Signaling with 1st Network
705
Signaling SW Signals 1st Network to Reach "Connected" State and then Locates 1st Real Interface Attaching to 1st Network
710
Connection Manager Configures 1st half-VSE to Associate w/ 1st Real Interface attaching to 1st Network by Creating a New Row in 1st Virtual Forwarding Table
715
Connection Manager Configures 2nd half-VSE for Signaling with 2nd Network
720
Signaling SW Signals 2nd Network to Reach "Connected" State and then Locates 2nd Real Interface Attaching to 2nd Network
725
Connection Manager Configures 2nd half-VSE to Associate w/ 2nd Real Interface attaching to 2nd Network by Creating a New Row in 2nd Virtual Forwarding Table
730
When Either Side Connection is Made (Reaches "Connected State"), Connection Manager Checks If Other Side is Already in the "Connected" State
740
If Yes, Connection Manager Cross-connects Two Identified Real Interfaces to build Data Path to Inter-connect Two Networks;
If No, Wait for Other Side to Reach "Connected" State

FIG 7 - Establishing a Connection - Gateway Networking Interworking Application

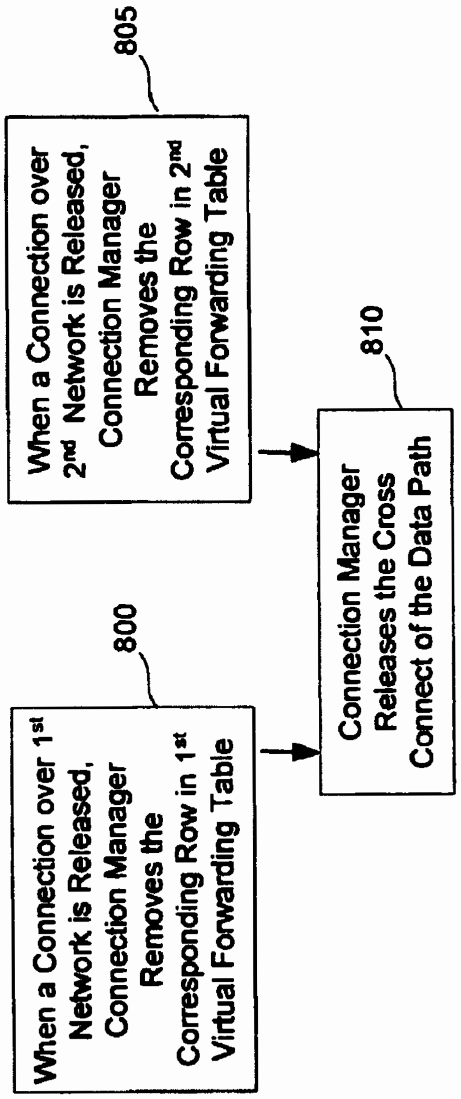

When a Connection over 1st Network is Released, Connection Manager Removes the Corresponding Row in 1st Virtual Forwarding Table
800
When a Connection over 2nd Network is Released, Connection Manager Removes the Corresponding Row in 2nd Virtual Forwarding Table
805
Connection Manager Releases the Cross Connect of the Data Path
810

FIG 8 - Releasing a Connection - Gateway Network Interworking Application

VIRTUAL SERVICE ENDPOINT

FIELD OF THE INVENTION

The present invention relates generally to network devices. More specifically, a virtual service endpoint is disclosed.

BACKGROUND OF THE INVENTION

A network switch may exchange data between multiple interfaces that run the same or different protocols: An interface is a physical module in a network switch which connects to customer premises equipment (CPE) or a network trunk, and runs a specific service or network protocol toward the customer side or network side.

FIG. 1A is a network diagram illustrating a service provider's wide-area ATM network **100** providing end-to-end services between customer premises equipment (CPE) or customer networks. The interfaces connecting the edge switches in a service provider's network and CPEs or customer networks define the service demarcation points between customers and the service provider. It is the service provider network's responsibility to interconnect two service demarcation points, or service endpoints, to allow communication between customer's different locations. As shown in FIG. 1A, network edge switches **120**, **122**, **124**, and **126** are each connected to an ATM/PNNI network **130**. Switch **120** is connected to a customer ATM network **132** and switch **122** is connected to a customer ATM CPE **134**. Switch **124** is connected to a metro Ethernet network **136** and Frame Relay network **138**. Switch **126** is connected to Ethernet network **140**, MPLS network **141**, and Frame Relay CPE **142**.

The service endpoints in switch **120** and switch **122** can communicate with each other via standards-based PNNI signaling and routing protocols. For example, for an ATM service endpoint in ATM network **132** to communicate with ATM CPE **134**, the ATM service endpoint in ATM network **132** sends PNNI standards-based signaling messages to set up a virtual circuit between the ATM service endpoint in ATM network **132** and the ATM service endpoint in ATM CPE **134**.

The service endpoints in switch **120** and switch **126** may not be able to communicate with each other because there is no standards based protocol for signaling and routing between service endpoints running different protocols. Note that the service endpoint in switch **120** is ATM, but the service endpoint in switch **126** is either Frame Relay or Ethernet. These may only be possible if switch **120** and switch **126** are from the same manufacturer, so they may develop some proprietary method of signaling between service endpoints with different protocol types. For example, for an ATM service endpoint in ATM network **132** to communicate with an Ethernet service endpoint in Ethernet network **140**, proprietary information may be included in a proprietary header in each packet sent to switch **126**

FIG. 1B is a packet format illustrating an ATM packet **150** with proprietary information carried in the packet header. Specifically, packet **150** includes standard ATM header information (**152**, **154**, and **158**) and proprietary information **162** to indicate, e.g., the real type of service endpoints. Packet **150** is an example of a packet with proprietary information that may be sent from an ATM service endpoint in ATM network **132** to an Ethernet service endpoint in Ethernet network **140**. Switch **126** uses proprietary information **162** to determine to which interface to send the packet for forwarding. The challenge of using proprietary header information or a proprietary encryption method is it requires that the edge switches that communicate with each other share the same proprietary

method to connect different types of service endpoints. In general, this means that participating edge switches must agree and understand the proprietary method beforehand, which most likely requires the two edge switches come from the same manufacturer, putting a significant constraint on service providers for introducing new edge switches to meet the need of network growth and expansion. There is a need to allow an edge switch to be inserted into an existing network running standard based signaling/routing protocols, to communicate between the service endpoints on existing edge switches and new service endpoints on the new edge switch which may not be recognizable by the standard signaling/routing protocols. For service providers, this would allow them to introduce new products into its existing network for new service deployment with minimum disruption. For switch vendors, it would also be beneficial to maintain a generic, common service interface in an edge switch, adaptive to the type of networking protocols running in the network the edge switch is connected to, to represent real service endpoints for signaling/routing over the network. This allows common software developed in the edge switch to connect to multiple networks running different networking protocols.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements, and in which:

FIG. 1A is a network diagram illustrating a service provider's wide-area ATM network **100** providing end-to-end services between customer premises equipment (CPE) or customer networks.

FIG. 1B is a packet format illustrating an ATM packet **150** with proprietary information carried in the packet header.

FIG. 2A is a block diagram illustrating a network **200** with edge switch **215**.

FIG. 2B is a block diagram illustrating a network **220** with a gateway switch **235**.

FIG. 3A is a diagram further illustrating edge switch **215** with a VSE **205** for performing edge service interworking.

FIG. 3B is a diagram, in principle, illustrating forwarding table **460** for edge switch **215**.

FIG. 4A is a diagram further illustrating gateway switch **235** inter-connecting ATM network **130** and MPLS network **141** via a VSE **225**.

FIG. 4B is a diagram, in principle, illustrating forwarding table **470** for gateway switch **235**.

FIG. 5 is a flowchart illustrating a method of establishing a connection for an edge switch for a service interworking application.

FIG. 6 is a flowchart illustrating a method of releasing a connection for an edge switch for a service interworking application.

FIG. 7 is a flowchart illustrating a method of establishing a connection for a gateway network interworking application.

FIG. 8 is a flowchart illustrating a method of releasing a connection for a gateway network interworking application.

DETAILED DESCRIPTION

It should be appreciated that the present invention can be implemented in numerous ways, including as a process, an apparatus, a system, or a computer readable medium such as a computer readable storage medium or a computer network wherein program instructions are sent over optical or electronic communication links. It should be noted that the order of the steps of disclosed processes may be altered within the scope of the invention.

A detailed description of one or more preferred embodiments of the invention is provided below along with accompanying figures that illustrate by way of example the principles of the invention. While the invention is described in connection with such embodiments, it should be understood that the invention is not limited to any embodiment. On the contrary, the scope of the invention is limited only by the appended claims and the invention encompasses numerous alternatives, modifications and equivalents. For the purpose of example, numerous specific details are set forth in the following description in order to provide a thorough understanding of the present invention. The present invention may be practiced according to the claims without some or all of these specific details. For the purpose of clarity, technical material that is known in the technical fields related to the invention has not been described in detail so that the present invention is not unnecessarily obscured.

A method of establishing a cross connection on a network switch between a first interface or service endpoint participating in a first protocol and a second interface or service endpoint participating in a second protocol is disclosed. The method comprises establishing a so-called Virtual Service Endpoint (VSE) in the "middle" of the switch, to participate in both the first protocol by connecting to the first interface/service endpoint and the second protocol by connecting to the second interface/service endpoint, and to facilitate setting up forwarding table to cross connect between the first interface/service endpoint and the second interface/service endpoint. A VSE is one-to-one correspondent to a cross-connect over the switch. A VSE is designed as an internal mechanism in the network switch, not to be exposed to or accessible from the outside world. In one embodiment, the first interface/service endpoint participates in a first protocol-specific network and the second interface/service endpoint participates in a second protocol-specific network. A VSE maintains a certain "duality" by presenting itself as a first protocol-specific service endpoint to the first protocol-specific network, and presenting itself as a second protocol-specific service endpoint to the second protocol-specific network. In principle, a virtual service endpoint can present any protocol-specific service endpoint recognizable by any protocol to allow communication between any two disparate networks running different network protocols.

Although a VSE corresponds to a cross-connect through a network switch, it can be further divided into two "paired" half-VSEs, which can be either identical or different. In one embodiment, a half-VSE is identified by the following structure of three fields: [Connection Type, Index 1, Index2], where each field is 32 bits long, Connection Type indicates what protocol-specific connection this half-VSE represents, and Index1 and Index2 represent the standard identifiers for the protocol-specific connection. As examples, a half-VSE representing an ATM virtual service endpoint has the following structure: [ATM VC, VPI, VCI], and a half-VSE representing an MPLS virtual service endpoint has the following structure: [MPLS LSP, forward MPLS Label, backward MPLS Label]. In one embodiment, the protocol translation method using Canonic Packet Format described in U.S. patent application Ser. No. 10/646,340 may be used together with the virtual service endpoint. In one such embodiment, the data format representing a virtual service endpoint is a canonical packet format.

A forwarding table is an element in a network switch that facilitates making a cross connect between an ingress interface and an egress interface. When a packet belonging to a virtual connection arrives at an ingress interface of a network switch, a forwarding table specifies what virtual connection the packet belongs to at egress, and what egress interface the packet exits. A forwarding table is typically constructed as a multiple-column table where each row represents a cross-connect over the switch, and the first half-column represents entries (connection type, connection identifiers, etc.) for the service endpoint at the ingress interface for the cross-connect, and the second half-column represents entries (connection type, connection identifiers, etc.) for the service endpoint at the egress interface for the cross-connect. While the forwarding table always specifies cross-connects between two "real" service endpoints at an ingress interface and an egress interface respectively, the notion of a VSE introduces the concept of "virtual" service endpoints, and a virtual forwarding table, to enable a cross-connect between any two disparate ingress and egress service interfaces attaching to disparate networks.

More specifically, the method can be used in a network device such as an edge switch enabling customer services or a gateway switch interconnecting two disparate networks running different network protocols. An edge switch connects a network (for example, connecting to a backbone network) to a service endpoint (for example, connecting to a customer network, or a CPE). A gateway switch connects two disparate networks (for example, a metro/regional ATM network and a backbone MPLS network). In one embodiment, the method is in used in a switch with a pooling switch, as further described in U.S. patent application Ser. No. 10/447,825.

FIG. **2A** is a block diagram illustrating a network **200** with edge switch **215**. Network **200** is similar to network **100** modified to include edge switch **215**, an Ethernet CPE **217** and a Frame Relay CPE **218**. Edge switch **215** connects ATM/PNNI network **130** with Ethernet CPE **217** and Frame Relay CPE **218**. Edge switch **215** includes a configured VSE **205** and forwarding table **210**.

Edge switches **122** and **215** can communicate via PNNI signaling/routing protocol to connect ATM CPE **134** and Ethernet CPE **217** because VSE **205**, associated with Ethernet CPE **217**, presents itself as an ATM service endpoint to ATM network **130**. Hence VSE **205** participates in PNNI signaling and routing protocols in ATM network **130**. For example, if switch **215** receives a (AAL5) packet sent from ATM CPE **134** to Ethernet CPE **217**, the packet arriving at the ATM trunk interface (attaching to ATM network **130**) is forwarded to Ethernet CPE **217** based on forwarding table **210**. The forwarding table **210** maps from the real ATM trunk interface to VSE **205**, and then from VSE **205** to the real Ethernet interface connecting to Ethernet CPE **217**.

FIG. **2B** is a block diagram illustrating a network **220** with a gateway switch **235**. Network **220** includes gateway switch **235** which connects ATM/PNNI network **130** with Ethernet network **140**, MPLS network **141**, and Frame Relay network **142**. Gateway switch **235** includes VSE **225** and forwarding table **230**.

Edge switch **120** and MPLS network **141** can communicate with each other via gateway switch **235** because VSE **225** presents itself (with its first half-VSE) as an ATM service endpoint to ATM network **130**, and in the meantime, presents itself (with its second half-VSE) as an MPLS service endpoint toward MPLS network **141**. For example, if gateway switch **235** receives a packet sent from an ATM endpoint on ATM network **132** to an MPLS service endpoint on MPLS network **141**, the packet is forwarded to MPLS network **141** based on forwarding table **230**. Forwarding table **230** forwards the packet from the real ingress interface (an ATM interface) attaching to ATM network **130** to first half-VSE of

225, and then from second half-VSE of 225 to the real egress interface (an MPLS/packet interface) attaching to MPLS network 141.

FIG. 3A is a diagram further illustrating edge switch 215 with a VSE 205 for performing edge service interworking. Edge switch 215 includes a first real interface 305 (which can be any type of service interface, e.g., Frame Relay, ATM, or Ethernet) and a second real interface 310, an ATM trunk interface, attaching to ATM network 130. VSE 205 includes a half-VSE 315 associated with first interface 305 and a half-VSE 320 associated with second interface 310. Half-VSE 315 presents itself to first interface 305 as an ATM interface, and half-VSE 320 presents itself to second interface 310 (and its attached ATM network 130) as an ATM service endpoint as well. Half-VSE 320 may participate in the PNNI signaling and routing protocols in the ATM network 130. Therefore, for providing an edge service interworking function, the two paired half-VSEs typically represent the same type of service endpoints: both half-VSEs 315 and 320 are virtual ATM service endpoints. In other embodiments, the virtual service endpoint may not be represented as two paired half-VSEs; it may be just one VSE.

FIG. 3B is a diagram, in principle, illustrating forwarding table 460 for edge switch 215. In this embodiment, edge switch 215 is configured to switch between first interface 305 which is a Frame Relay interface, and second interface 310 which is an ATM interface attached to ATM network 130. Forwarding table 460 includes two "virtual" forwarding tables 400 and 405, combined to form real forwarding table 460. Note that the two half virtual interfaces 412 and 414 are identical. Each row in the virtual forwarding table 400 specifies the association between a real service endpoint on Frame Relay interface 305 and a first half-VSE which is an ATM interface. Each row in the virtual forwarding table 405 specifies the association between a second half-VSE which is also an ATM interface and a real service endpoint on ATM interface 310. Each row in each virtual forwarding table associates a real service endpoint (in a real interface) with a half virtual service endpoint. Combining the two virtual forwarding tables 400 and 405 results in the real forwarding table which cross-connects between the Frame Relay interface 305 and the ATM interface 310.

Each row in the virtual forwarding table 400 includes entries for a first real service endpoint in first real Frame Relay interface 305 and entries for a first half-VSE. Entries of first half-VSE include the identifiers that are presented to first real Frame Relay interface 305. For example, if edge switch 215 receives a packet with an identifier [FR Connection ID1, DLCI1], the identifier for its associated first half-VSE is [ATM Connection ID1, VPI1, VCI1]. Identifier [ATM Connection ID1, VPI1, VCI1] is recognizable by ATM network 130, hence is able to participate in the PNNI signaling and routing in ATM network 130. As far as ATM network 130 can tell, the edge switch 215 and beyond is just ATM.

Each row in the virtual forwarding table 400 includes entries for a first real service endpoint in real interface 305 and entries for a first half-VSE. Entries of first half-VSE include the identifiers that are presented to first real interface 305. For example, if edge switch 215 receives a FR packet with an identifier [FR Connection ID1, DLCI1], the identifier for its associated first half-VSE is [ATM Connection ID1, VPI1, VCI1], and its paired second half-VSE also has the identifier [ATM Connection ID1, VPI1, VCI1], which is recognizable by ATM network 130, hence is able to participate in the PNNI signaling and routing in ATM network 130. As far as ATM network 130 can tell, edge switch 215 and all its service endpoints are just ATM. The arriving FR packet with identifier [FR Connection ID1, DLCI1] is forwarded to ATM network 130 with identifier [ATM Connection ID1, VPI4, VCI4].

FIG. 4A is a diagram further illustrating gateway switch 235 inter-connecting ATM network 130 and MPLS network 141 via a VSE 225. Gateway switch 235 includes a first real interface 335, an ATM interface, attaching to ATM network 130, and a second real interface 340, an MPLS packet interface, attaching to MPLS network 141. Virtual service endpoint 225 includes a half-VSE 345 associated with first real interface 335 and its attached ATM network 130, and a half-VSE 350 associated with second real interface 340 and its attached MPLS network 141. Half-VSE 345 presents itself to first interface 335 (and ATM network 130) as an ATM service endpoint, and half-VSE 350 presents itself to second interface 340 (and MPLS network 141) as an MPLS service endpoint. First half-VSE 345 is viewed as an ATM service endpoint participating in PNNI routing and signaling protocols in ATM network 130, while second half-VSE 350 is viewed as an MPLS service endpoint participating in MPLS routing and signaling protocols in MPLS network 141. Therefore, for providing gateway network interworking function, the two paired half-VSEs typically represent two different types of service endpoints: half-VSE 345 is a virtual ATM service endpoint, but half-VSE 350 is a virtual MPLS service endpoint. In other embodiments, the virtual service endpoint may not be represented as two half virtual service endpoints. The virtual service endpoint may be represented as two virtual service endpoints or represented in other ways to provide the functionality described herein.

FIG. 4B is a diagram, in principle, illustrating forwarding table 470 for gateway switch 235. Gateway switch 235 is attached to both ATM network 130 and MPLS network 141. Forwarding table 470 includes two "virtual" forwarding tables 420 and 425. In this embodiment, virtual forwarding table 420 specifies the association between three real service endpoints in first real ATM interface 305 (attaching to ATM network 130) and three (first) half-VSEs in gateway switch 235. The virtual forwarding table 425 specifies the association between three (second) half-VSEs and three real MPLS service endpoints in second real MPLS interface 340 (attaching to MPLS network 141). Each row in each virtual forwarding table associates a real service endpoint (in a real interface) with a half virtual service endpoint. Combining two virtual forwarding tables 420 and 425 results in the real forwarding table 470 which connects the real service endpoints in ATM interface 335 to the real service endpoints in MPLS interface 340, hence interconnects ATM network 130 and MPLS network 141.

Each row in the virtual forwarding table 420 includes entries for a first real service endpoint in first real ATM interface 335 and entries for a first half-VSE. Entries of first half-VSE include the identifiers that are presented to first real interface 335. For example, if gateway switch 235 receives a packet with an identifier [ATM Connection ID1, VPI1, VCI1], the identifier for its associated first half-VSE is [ATM Connection ID4, VPI4, VCI4]. Identifier [ATM Connection ID4, VPI4, VCI4] is recognizable by ATM network 130, hence is able to participate in the PNNI signaling and routing in ATM network 130. As far as ATM network 130 can tell, the gateway switch 235 and beyond is just ATM.

Each row in the virtual forwarding table 425 includes entries for a second real service endpoint in real interface 340 and entries for a second half-VSE. Entries of second half-VSE include the identifiers that are presented to second real interface 340. For example, if gateway switch 235 receives a (AAL5) packet with an identifier [ATM Connection ID1, VPI1, VCI1], the identifier for its associated second half-VSE is [MPLS Connection ID1, VC Label1 (Forward Direction), VC Label1 (Reverse Direction)], which is recognizable by MPLS network **141**, hence is able to participate in the MPLS signaling and routing in MPLS network **141**. As far as MPLS network **141** can tell, gateway switch **235** and beyond is just MPLS. The arriving (AAL5) packet with an identifier [ATM Connection ID1, VPI1, VCI1] from the ATM network **130** is forwarded to MPLS network **141** with identifier [MPLS Connection ID4, VC Label2 (Forward Direction), VC Label2 (Reverse Direction)].

FIG. **5** is a flowchart illustrating a method of establishing a connection for an edge switch for a service interworking application. A piece of software called connection manager in the switch is responsible for building a forwarding table for cross-connect over the switch. The connection manager maintains all the configuration information and keeps track of all connections and status. For establishing a circuit over the switch and attached network, two half-VSEs are configured in the switch, where the first half-VSE is configured to associate with the first real service endpoint by creating a new row in first virtual forwarding table (step **500**), and the second half-VSE is configured to be associated with the first half-VSE (step **510**). Network signaling SW participates in network signaling/routing (e.g., PNNI signaling for ATM network) using second half-VSE as service endpoint (step **520**). When networking signaling reaches the "connected" state, the second real interface is located (the module where all signaling calls going through). The network signaling software sends a request to the connection manager to connect the second half-VSE with the second real interface by creating a new row in the second virtual forwarding table (step **530**). Because of the association between the first half-VSE and the second half-VSE, the connection manager identifies the two rows in the two virtual forwarding tables, and make the association between the identified first real service endpoint and second real interface (step **540**). The connection manager finally cross-connects the two resulting real interfaces to establish the data path through the switch (step **550**).

FIG. **6** is a flowchart illustrating a method of releasing a connection for an edge switch for a service interworking application. Signaling software requests that second half-VSE is disconnected from the second real interface and the attached network (step **600**). The connection manager removes second half-VSE's corresponding row in second virtual forwarding table (step **610**). Based on the association between the first half-VSE and the second half-VSE, the connection manager locates first half-VSE in first virtual forwarding table (step **620**), and then finds the first real service endpoint for the connection (step **630**). The connection manager removes the corresponding row in the first virtual forwarding table and disconnects the data path between the two real service endpoints (step **640**).

FIG. **7** is a flowchart illustrating a method of establishing a connection for a gateway network interworking application. A gateway switch connects two networks, the first network and the second network. The connection to each network is established independently. In one embodiment, each connection is established according to steps **500**-**550**. A connection with the first network is established between the first half-VSE and first real interface attached to the first network (steps **700**, **705**, **710**). For example, if the first network is an ATM network, the connection is established as either a source or destination SPVC, as if the SPVC was terminated to a fixed service endpoint (e.g., Frame Relay or ATM service endpoint) on the gateway switch. Similarly, a connection with the second network is established between the second half-VSE and second real interface attached to the second network (steps **715**, **720**, **725**). For example, if the second network is an MPLS network, the connection is established by LDP, using Martini VC for establishing connections over MPLS tunnels, again as if the Martini connection was terminated to a fixed service endpoint (e.g., Frame Relay or ATM service endpoint) on the gateway switch. The connection manager is informed of the status of two connections from signaling software which signals two connections toward two networks, respectively. When either connection is made (i.e., when the corresponding call reaches the "connected" state), the connection manager checks whether the other side is in the "connected" state (step **730**). If yes, the connection manager cross-connects two identified real interfaces to establish a data path over the gateway switch to inter-connect the two connections through two networks; if no, the connection manager waits for the other side to reach the "connected" state (step **740**).

FIG. **8** is a flowchart illustrating a method of releasing a connection for a gateway network interworking application. When either connection is released, the connection manager releases the cross-connect of the data path over the gateway switch (steps **800**, **805**, **810**). Each side of the connection handles the failure of the connection in its respective network. For example, for an ATM network, the failure of the connection is handled like a transmission failure on a PVC. The call remains in the "connected" state from a network signaling point of view. OAM F5 cells are transmitted by the switch to indicate that the connection is not available. For an MPLS network, the LDP label of the Martini VC would be withdrawn, resulting in the release of the VC. These conditions are removed when the connection is reestablished.

Other embodiments of the methods described above may perform different and/or additional steps. In addition, the steps may be performed in different orders.

Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. It should be noted that there are many alternative ways of implementing both the process and apparatus of the present invention. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.

What is claimed is:

1. A method of establishing a connection between a first endpoint participating in a first protocol and a second endpoint participating in a second protocol, the method including:

establishing a virtual service endpoint to participate in the first protocol on a network device coupled to the first endpoint and the second endpoint, wherein the virtual service endpoint is configured to provide interworking between the first protocol and the second protocol, wherein the virtual service endpoint includes a first half virtual service endpoint and a second half virtual service endpoint, the first half virtual service endpoint to participate in the first protocol and the second half virtual service endpoint to participate in the second protocol; and

using a forwarding table to forward communication to the second endpoint via the virtual service endpoint, the forwarding table to include:

a mapping from a first interface associated with the first protocol to the first half virtual service endpoint;

a mapping from the first half virtual service endpoint to the second half virtual service endpoint; and a mapping from the second half virtual service endpoint to a second interface associated with the second protocol.

2. A method as recited in claim 1, wherein the first interface associated with the first protocol comprises a first real interface for the first endpoint and the second interface associated with the second protocol comprises a second real interface for the second endpoint.

3. A method as recited in claim 1, wherein using the forwarding table to forward communication to the second endpoint includes:

coupling the first half virtual service endpoint to the first interface associated with the first protocol;

locating the first half virtual service endpoint in the forwarding table and finding the second half virtual service endpoint; and

coupling the second half virtual service endpoint to the second interface associated with the second protocol, wherein coupling the first half and the second half virtual service endpoints to the first interface and the second interface couples a data path between the first endpoint and the second endpoint.

4. A method as recited in claim 1, wherein to forward communication to the second endpoint includes:

establishing a connection between the first interface and the first half virtual service endpoint;

establishing a connection between the second interface and the second half virtual service endpoint;

and cross connecting a data path between the two connections.

5. A method as recited in claim 1, wherein the first half virtual service endpoint has an identifier associated with the first interface and the second half virtual service endpoint has an identifier associated with the second interface.

6. A method as recited in claim 1, wherein the first protocol or the second protocol are associated with one of ATM, Ethernet, Frame Relay, or MPLS.

7. A method as recited in claim 1, wherein the network device is an edge switch or a gateway switch.

8. A method as recited in claim 1, wherein the first half virtual service endpoint and the second half virtual service endpoint are identical.

9. A method as recited in claim 1, wherein the first endpoint is not aware of the second protocol and the second endpoint is not aware of the first protocol.

10. A method as recited in claim 1, wherein:

the virtual service endpoint includes the first endpoint and the second endpoint;

the first endpoint includes the first interface and the first half virtual service endpoint; and

the second endpoint includes the second interface and the second half virtual service endpoint.

11. A method as recited in claim 10, wherein the virtual service endpoint is located on the network device.

12. A method as recited in claim 1, wherein the virtual service endpoint is located on the network device.

13. A network device for establishing a connection between a first endpoint participating in a first protocol-specific network and a second endpoint participating in a second protocol-specific network, the network device including:

a virtual service endpoint configured to:

provide interworking between the first protocol-specific network and the second protocol-specific network, wherein the virtual service endpoint is to present a first protocol-specific service endpoint to the first protocol-specific network and present a second protocol-specific service endpoint to the second protocol-specific network; and

use a forwarding table to forward communication from the first endpoint to the second endpoint through the virtual service endpoint, the forwarding table to include:

a mapping from a first interface associated with the first protocol-specific network to the first protocol-specific network;

a mapping from the first protocol-specific endpoint to the second protocol-specific endpoint; and

a mapping from the second protocol-specific endpoint to a second interface associated with the second protocol-specific network;

wherein the network device is connected to coupled with the first endpoint and the second endpoint.

14. A network device as recited in claim 13, wherein the first protocol-specific network or the second protocol-specific network are associated with one of ATM, Ethernet, Frame Relay, or MPLS.

15. A network device as recited in claim 13, wherein to use the forwarding table to forward communication from the first endpoint to the second endpoint includes the virtual service endpoint further configured to:

establish a connection between the first interface and the first protocol-specific service endpoint;

establish a connection between the second interface and the second protocol-specific service endpoint;

and cross connect a data path between the two connections.

16. A computer program product for establishing a connection between a first endpoint participating in a first protocol and a second endpoint participating in a second protocol, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions to:

establish a virtual service endpoint to participate in the first protocol on a network device coupled to the first endpoint and the second endpoint, wherein the virtual service endpoint is configured to provide interworking between the first protocol and the second protocol, wherein the virtual service endpoint includes a first half virtual service endpoint and a second half virtual service endpoint, the first half virtual service endpoint to participate in the first protocol and the second half virtual service endpoint to participate in the second protocol; and

use a forwarding table to forward communication to the second endpoint via the virtual service endpoint, the forwarding table to include:

a mapping from a first interface associated with the first protocol to the first half virtual service endpoint;

a mapping from the first half virtual service endpoint to the second half virtual service endpoint; and

a mapping from the second half virtual service endpoint to a second interface associated with the second protocol.

17. A computer program product as recited in claim 16, wherein the first protocol or the second protocol are associated with one of ATM, Ethernet, Frame Relay, or MPLS.

18. A computer program product as recited in claim 16, wherein to use the forwarding table to forward communication to the second endpoint includes instructions to:

couple the first half virtual service endpoint to the first interface associated with the first protocol;

locate the first half virtual service endpoint in the forwarding table and find the second half virtual service endpoint; and

couple the second half virtual service endpoint to the second interface associated with the second protocol, wherein to couple the first half and the second half virtual service endpoints to the first interface and the second interface couples a data path between the first endpoint and the second endpoint.

19. A method of establishing a connection between a first endpoint participating in a first protocol-specific network and a second endpoint participating in a second protocol-specific network, the method including:

establishing a virtual service endpoint with a network device coupled with the first endpoint and the second endpoint, the virtual service endpoint configured to provide interworking between the first protocol-specific network and the second protocol-specific network, wherein the virtual service endpoint presents a first protocol-specific service endpoint to the first protocol-specific network and presents a second protocol-specific service endpoint to the second protocol-specific network; and

using a forwarding table to forward communication from the first endpoint to the second endpoint via the virtual service endpoint, the forwarding table to include:

a mapping from a first interface associated with the first protocol-specific network to the first protocol-specific service endpoint;

a mapping from the first protocol-specific service endpoint to the second protocol-specific service endpoint; and

a mapping from the second protocol-specific service endpoint to a second interface associated with the second protocol-specific network.

20. A method as recited in claim **19**, wherein the first protocol-specific network or the second protocol-specific network are associated with one of ATM, Ethernet, Frame Relay, or MPLS.

21. A method as recited in claim **19**, wherein using the forwarding table to forward communication from the first endpoint to the second endpoint includes:

establishing a connection between the first interface and the first protocol-specific service endpoint;

establishing a connection between the second interface and the second protocol-specific service endpoint;

and cross connecting a data path between the two connections.

* * * * *

Exhibit D

US007940652B1

(12) **United States Patent**
Pan

(10) **Patent No.:** **US 7,940,652 B1**
(45) **Date of Patent:** **May 10, 2011**

(54) **PSEUDOWIRE PROTECTION USING A STANDBY PSEUDOWIRE**

(75) Inventor: **Ping Pan**, San Jose, CA (US)

(73) Assignee: **Brixham Solutions Ltd.**, Tortola (VG)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 861 days.

(21) Appl. No.: **11/354,569**

(22) Filed: **Feb. 14, 2006**

Related U.S. Application Data

(60) Provisional application No. 60/653,065, filed on Feb. 14, 2005.

(51) **Int. Cl.**
H04J 3/14 (2006.01)

(52) **U.S. Cl.** **370/228**; 370/216; 370/225; 709/220

(58) **Field of Classification Search** 370/216, 370/225, 228; 709/220
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,920,705	A	7/1999	Lyon et al.	
6,167,051	A	12/2000	Nagami et al.	
6,347,088	B1	2/2002	Katou et al.	
6,430,184	B1	8/2002	Robins et al.	
6,546,427	B1	4/2003	Ehrlich	
6,574,477	B1 *	6/2003	Rathunde	455/453
6,621,793	B2	9/2003	Widegren et al.	
6,665,273	B1	12/2003	Goguen et al.	
6,680,943	B1	1/2004	Gibson et al.	
6,751,684	B2	6/2004	Owen et al.	
6,813,271	B1	11/2004	Cable	
6,845,389	B1	1/2005	Sen et al.	
6,985,488	B2	1/2006	Pan et al.	
7,050,396	B1	5/2006	Cohen et al.	
7,200,104	B2 *	4/2007	Saleh et al.	370/216
7,436,782	B2	10/2008	Ngo et al.	
7,697,528	B2	4/2010	Parry	
2001/0021175	A1	9/2001	Haverinen	
2001/0023453	A1	9/2001	Sundqvist	
2002/0112072	A1	8/2002	Jain	
2002/0141393	A1	10/2002	Eriksson	
2002/0146026	A1	10/2002	Unitt et al.	
2003/0002482	A1	1/2003	Kubler et al.	
2003/0039237	A1	2/2003	Forslow	
2003/0117950	A1 *	6/2003	Huang	370/220
2004/0105459	A1	6/2004	Mannam	
2004/0114595	A1	6/2004	Doukai	
2004/0133692	A1 *	7/2004	Blanchet et al.	709/230
2004/0156313	A1	8/2004	Hofmeister	
2004/0174865	A1	9/2004	O'Neill	
2004/0252717	A1	12/2004	Solomon et al.	

(Continued)

OTHER PUBLICATIONS

Ziying Chen: "The LSP Protection/Restoration Mechanism in GMPLS" Internet Citatio (Online) Oct. 2002 (Oct. 1, 2002), XP002239552 Retrieved from the ineternet URL: http://www.site.uottawa.ca/~bochmann/dsrg/PublicDocuments/Master-theses/Chen,%20Ziying%20%20-%202002.pdf.*

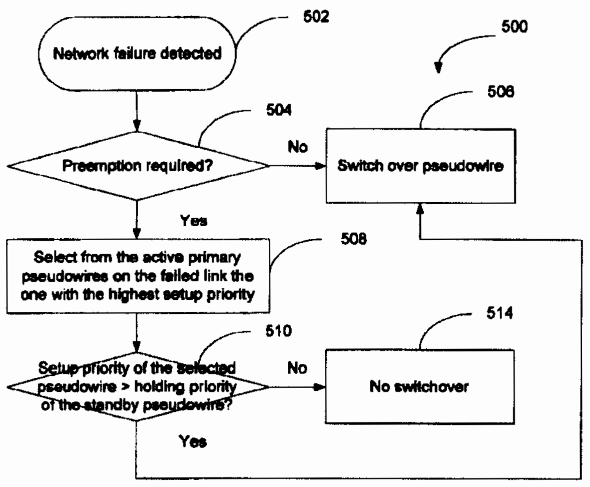
(Continued)

Primary Examiner — William Trost, IV
Assistant Examiner — Siming Liu

(57) **ABSTRACT**

Providing protection to network traffic includes sending a Pseudowire protection configuration parameter for configuring a standby Pseudowire between a source node and a destination node, receiving a Pseudowire configuration acknowledgement indicating whether the Pseudowire protection configuration parameter has been accepted by the destination node, and in the event that the Pseudowire protection configuration parameter has been accepted by the destination node, using the standby Pseudowire, wherein the standby Pseudowire is configured based at least in part on the Pseudowire protection configuration parameter.

17 Claims, 7 Drawing Sheets


500
502
Network failure detected
504
Preemption required?
No
506
Switch over pseudowire
Yes
508
Select from the active primary pseudowires on the failed link the one with the highest setup priority
510
Setup priority of the selected pseudowire > holding priority of the standby pseudowire?
No
514
No switchover
Yes

U.S. PATENT DOCUMENTS

2005/0018605	A1	1/2005	Foote	
2005/0044262	A1	2/2005	Luo	
2005/0220148	A1	10/2005	DelRegno	
2005/0237927	A1	10/2005	Kano et al.	
2006/0002423	A1	1/2006	Rembert	
2006/0018252	A1 *	1/2006	Sridhar et al.	370/216
2006/0046658	A1 *	3/2006	Cruz et al.	455/67.11
2006/0047851	A1 *	3/2006	Voit et al.	709/239
2006/0090008	A1	4/2006	Guichard	
2006/0146832	A1	7/2006	Rampal	
2006/0233167	A1	10/2006	McAllister	
2007/0053366	A1	3/2007	Booth, III	
2007/0127479	A1	6/2007	Sinicrope et al.	
2007/0206607	A1	9/2007	Chapman	
2008/0031129	A1	2/2008	Arseneault	

OTHER PUBLICATIONS

Braden, R. et al., "Integrated Services in the Internet Architecture: an overview," Network Working Group, Jun. 1994.

Bryant, S. et al., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture," Network Working Group, Mar. 2005.

Blake, S. et al., "An Architecture for Differentiated Services," Network Working Group, Dec. 1998.

Shah, Himanshu et al., Internet Draft, ARP Mediation for IP Interworking of Layer 2 VPN, L2VPN Working Group, Jul. 2007.

Martini, Luca et al., Internet Draft, Segmented Pseudo Wire, Network Working Group, Jul. 2007.

Pan, P. et al., Internet Draft, Pseudo Wire Protection, Jul. 2006.

Rosen, Eric C. et al., Internet Draft, PWE3 Congestion Control Framework, Network Working Group, Mar. 2004.

Rosen, E. et al., BGO-MPLS IP Virtual Private Networks (VPN), Network Working Group, Feb. 2006.

Pan, Ping, Internet Draft, Dry-Martini: Supporting Pseudo-wires in Sub-IP Access Networks, Network Working Group, Jul. 2005.

Mcpherson et al., Pseudowire Emulation Edge to Edge (PWE3) Jun. 13, 2007, http://www.ietforg/html.charters/pwe3-carter.html.

Afferton, Thomas S. et al., Ethernet Transport over Wide Area Networks, Packet-Aware Transport for Metro Networks, IEEE Communications Magazine, pp. 120-127, Mar. 2004.

Martini, L. et al., Pseudowire Setup and Maintenance using the Label Distribution Protocol (LDP), Network Working Group, Apr. 2006.

Anderson, L. et al., LDP Specification, Network Working Group, Jan. 2001.

Martini, Luca et al., Encapsulation Methods for Transport of Ethernet over MPLS Networks, Network Working Group, Apr. 2006.

Martini, Luca et al., Encapsulation Methods for Transport of Frame Relay Over MPLS Networks, Network Working Group, Feb. 2006.

Metz, Chris et al., Pseudowire Attachment Identifiers for Aggregation and VPN Autodiscovery, PWE3 Working Group, Feb. 25, 2006.

Martini, Luca et al., Dynamic Placement of Multi Segment Pseudo Wires, PWE3 Working Group, Jun. 2006.

Martini, Luca et al., "Pseudowire Setup and Maintenance using LDP", Network Working Group, Mar. 2005.

Vasseur, et al., Path Computation Element (pce), May 9, 2007, http://www.ietforg/html.charters/pce.charter.html.

Theimer, T. et al, "Requirements for OAM Functionality in MPLS", Oct. 1999, Watersprings.

Harry Newton, "Newton's Telecom Dictionary", 23rd Updated and Expanded Edition, p. 825,p. 239, Flatiron Publishing, New York, Mar. 2007.

* cited by examiner

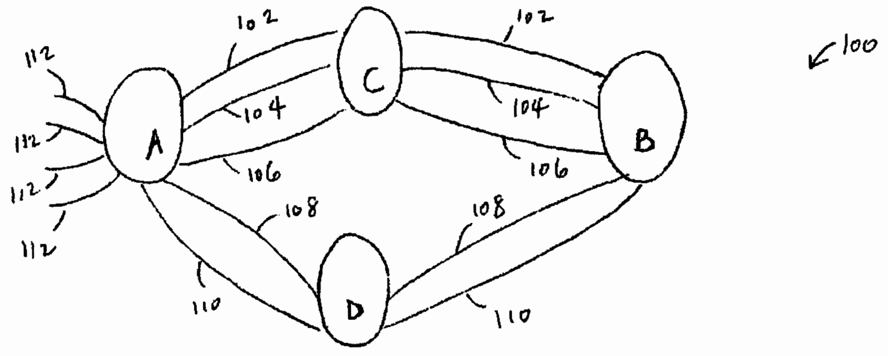

100
112
112
112
112
A
102
104
106
108
110
C
102
104
106
B
108
110
D

FIG. 1A

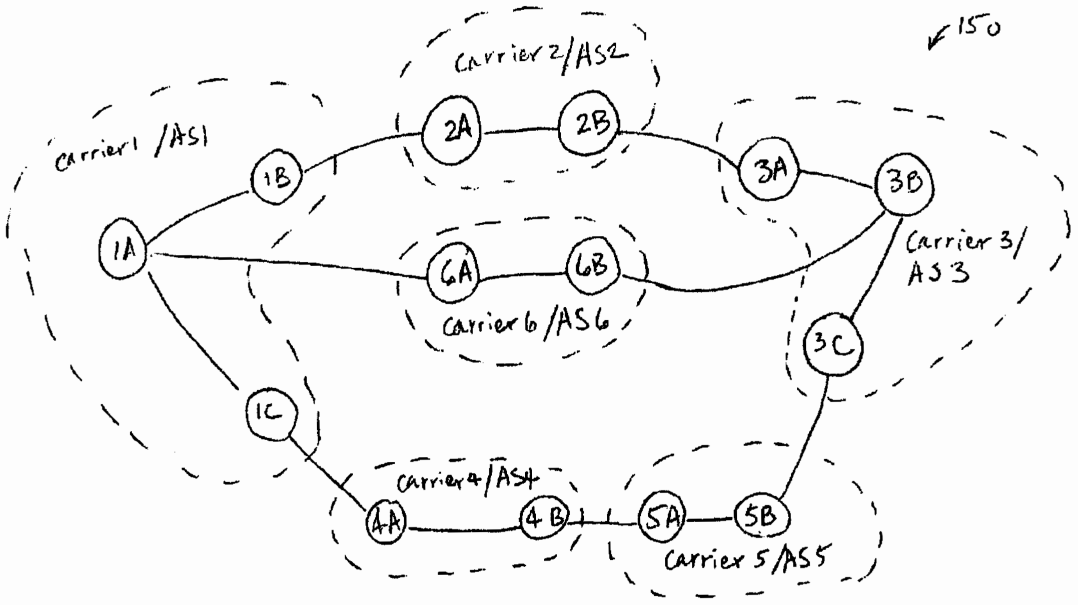

150
Carrier 1 /AS1
1A
1B
1C
Carrier 2/AS2
2A
2B
Carrier 3/AS3
3A
3B
3C
6A
6B
Carrier 6 /AS6
Carrier 4/AS4
4A
4B
5A
5B
Carrier 5/AS5

FIG. 1B

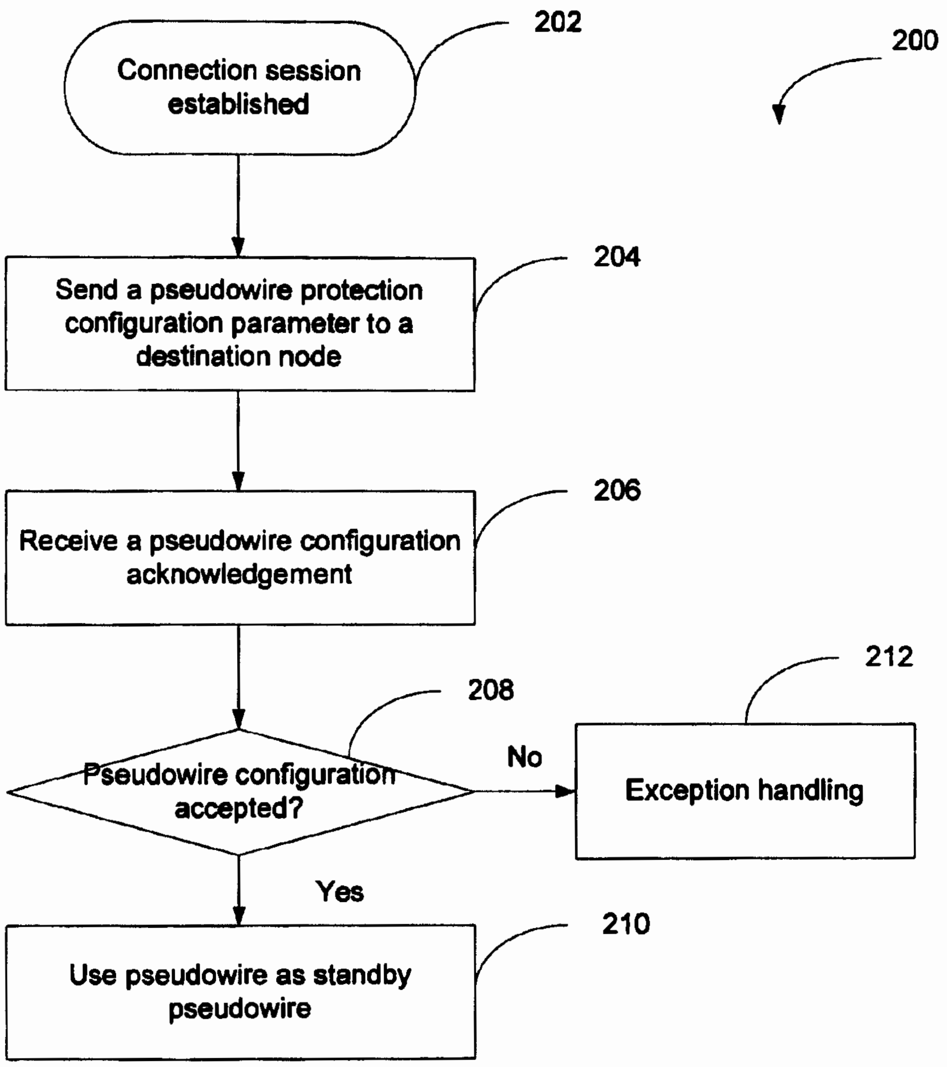

200
Connection session established
202
Send a pseudowire protection configuration parameter to a destination node
204
Receive a pseudowire configuration acknowledgement
206
Pseudowire configuration accepted?
208
No
Exception handling
212
Yes
Use pseudowire as standby pseudowire
210

FIG. 2

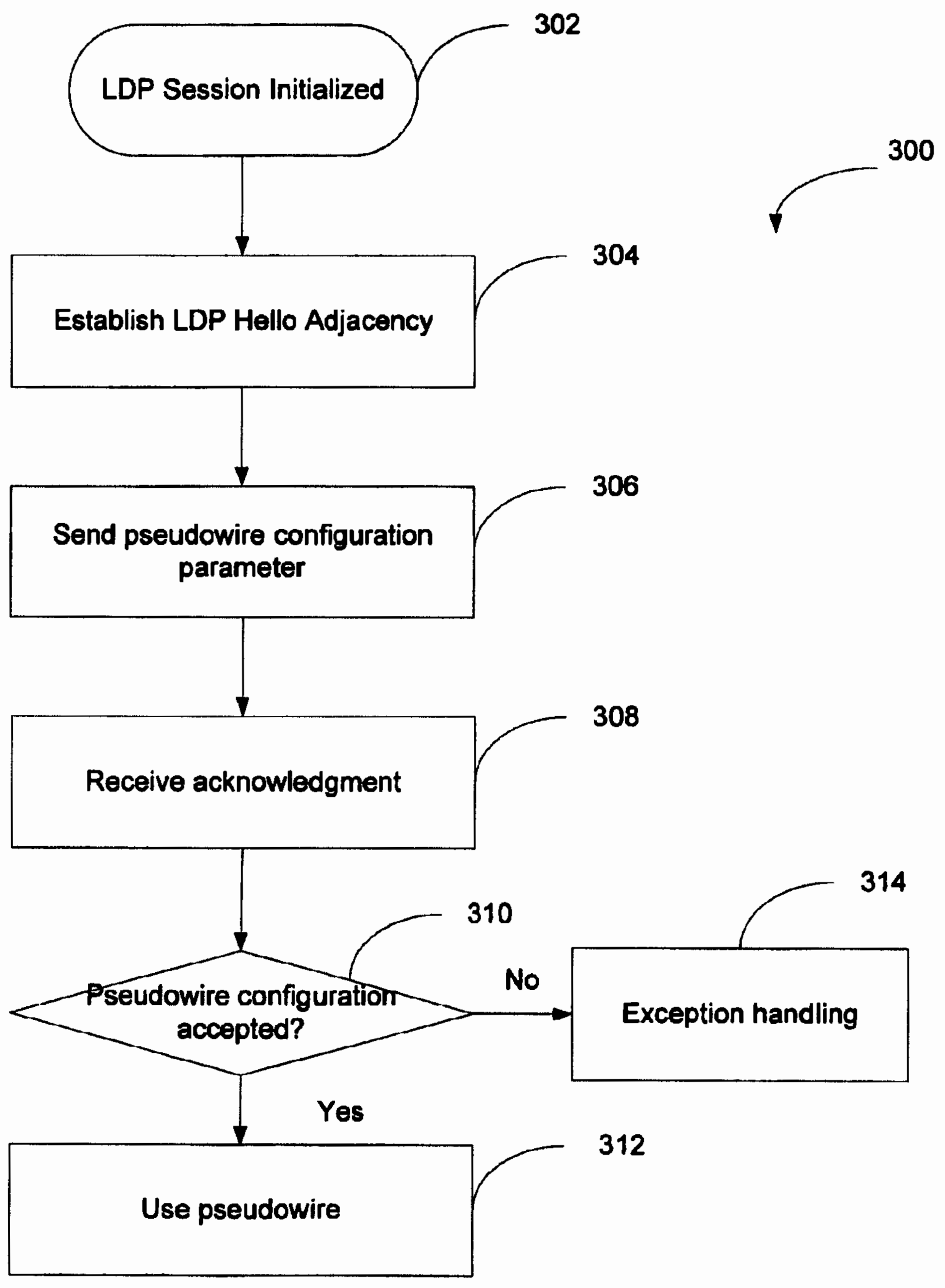

302
LDP Session Initialized
300
304
Establish LDP Hello Adjacency
306
Send pseudowire configuration parameter
308
Receive acknowledgment
310
314
Pseudowire configuration accepted?
No
Exception handling
Yes
312
Use pseudowire

FIG. 3A

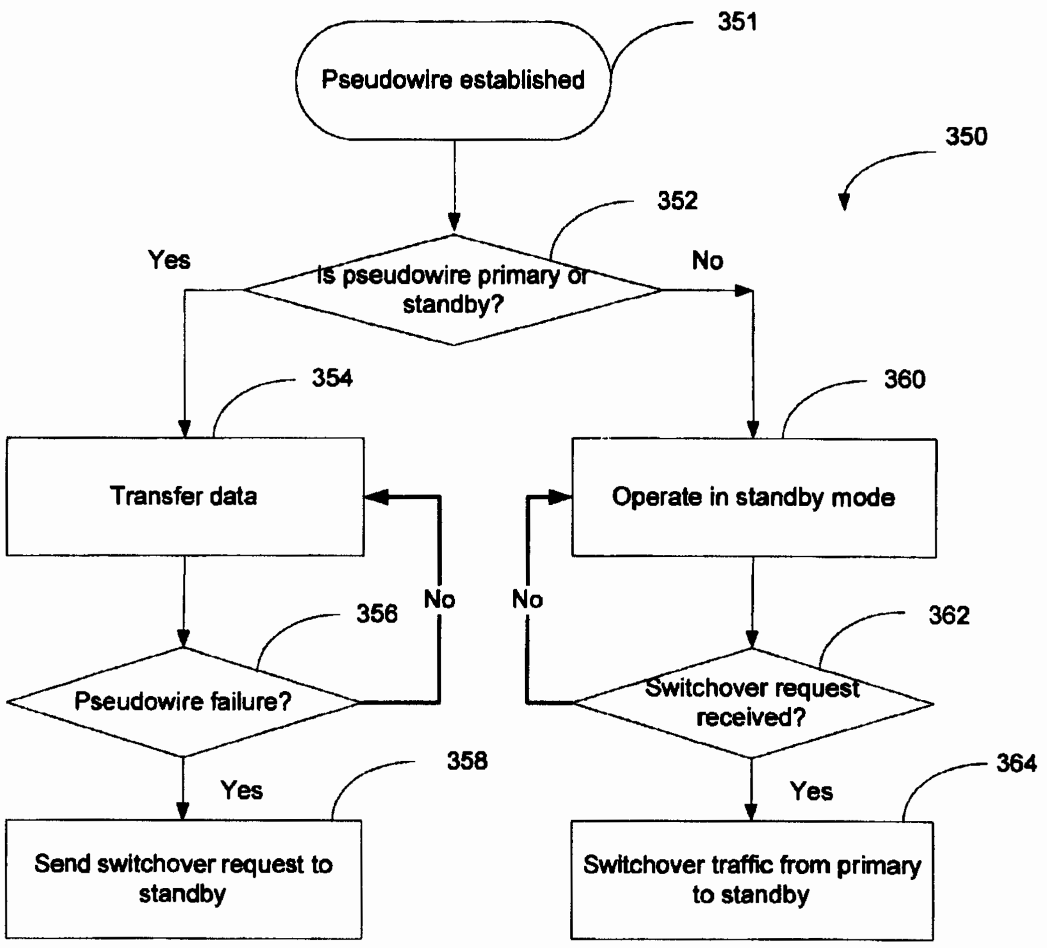

351
Pseudowire established
350
352
Is pseudowire primary or standby?
Yes
No
354
Transfer data
360
Operate in standby mode
356
No
No
Pseudowire failure?
362
Switchover request received?
358
Yes
364
Yes
Send switchover request to standby
Switchover traffic from primary to standby

FIG. 3B

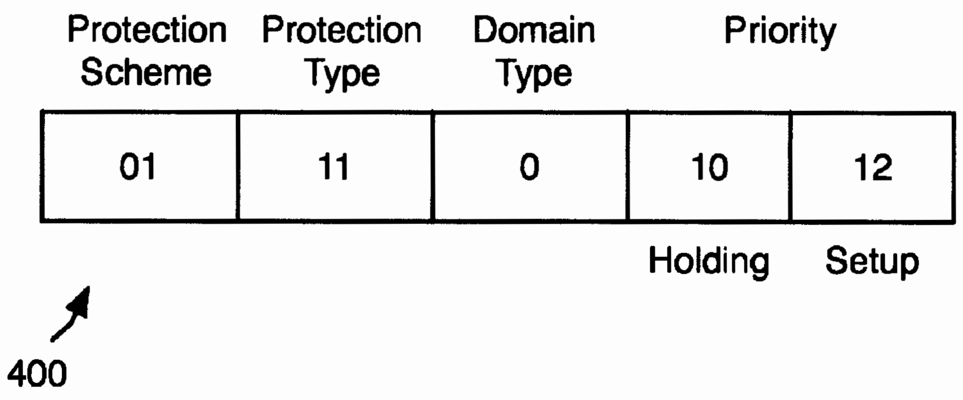

Protection Scheme
Protection Type
Domain Type
Priority
01
11
0
10
12
Holding
Setup
400

FIG. 4

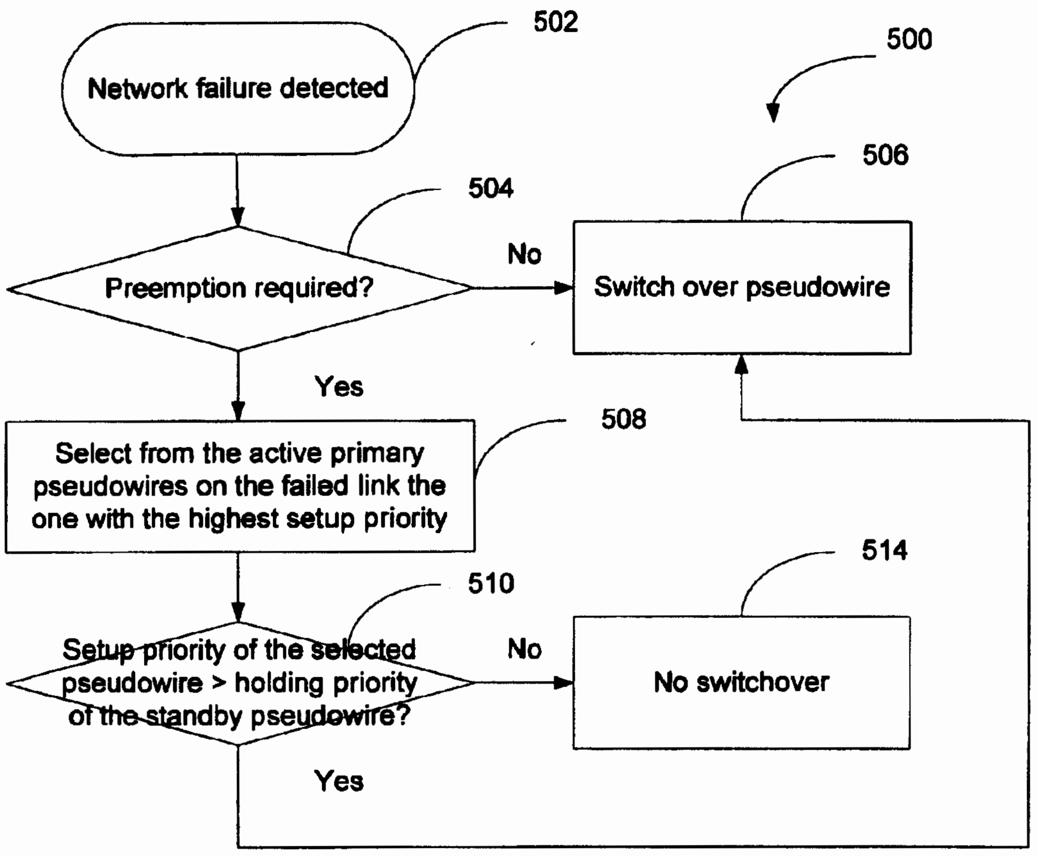

502
500
Network failure detected
506
504
No
Preemption required?
Switch over pseudowire
Yes
508
Select from the active primary pseudowires on the failed link the one with the highest setup priority
514
510
No
Setup priority of the selected pseudowire > holding priority of the standby pseudowire?
No switchover
Yes

FIG. 5

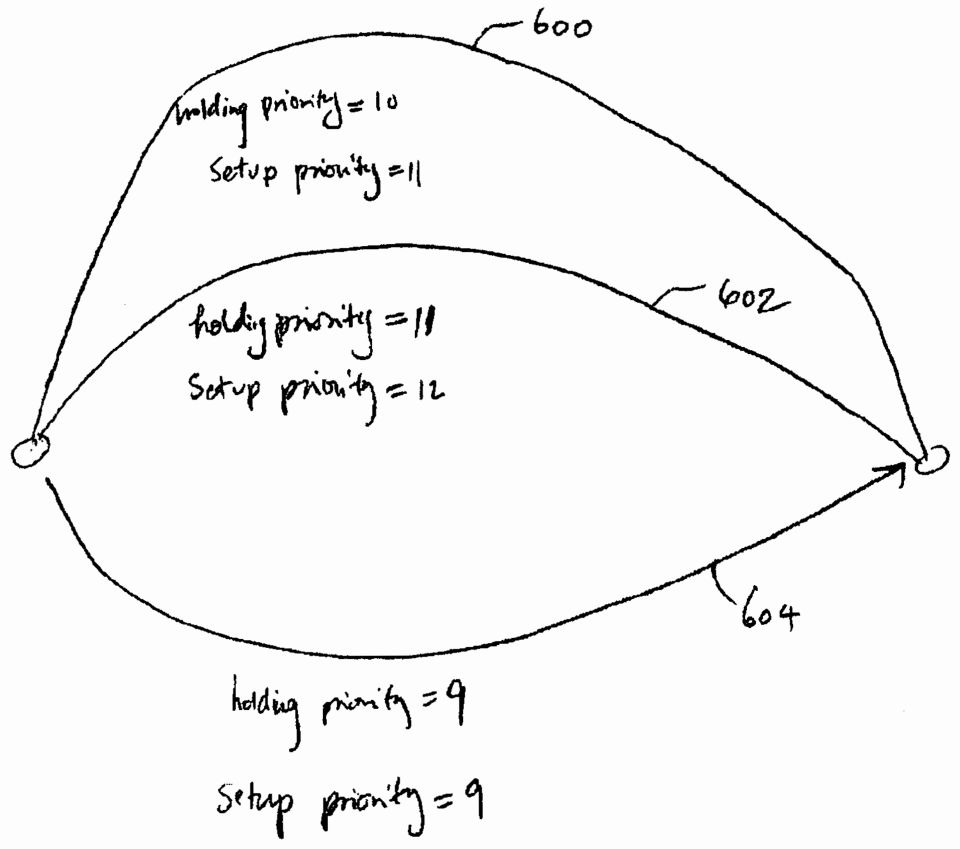

600
holding priority = 10
Setup priority = 11
602
holding priority = 11
Setup priority = 12
604
holding priority = 9
Setup priority = 9

FIG. 6

PSEUDOWIRE PROTECTION USING A STANDBY PSEUDOWIRE

CROSS REFERENCE TO OTHER APPLICATIONS

This application claims priority to U.S. Provisional Patent Application No. 60/653,065 entitled PSEUDO WIRE PROTECTION filed Feb. 14, 2005 which is incorporated herein by reference for all purposes.

BACKGROUND OF THE INVENTION

In recent years, many networking and telecommunications carriers have deployed Pseudowires to carry Layer-2 (also known as the data link layer of the Open Systems Interconnection (OSI) Reference Model) traffic. A Pseudowire (PW) refers to an emulation of a native service over a network. Examples of the native service include Asynchronous Transfer Mode (ATM), Frame Relay, Ethernet, Time Division Multiplexing (TDM), Synchronous Optical Network (SONET), Synchronous Digital Hierarchy (SDH), etc. Examples of the network include Multiprotocol Label Switching (MPLS), Internet Protocol (IP), etc. More recently, a number of carriers have extended the use of Pseudowires beyond packet encapsulation, and offered Pseudowires as a type of network service. Consequently, data traffic protection and redundancy in environments that use Pseudowire have become critical.

At the edge of a network, a network edge device such as an edge router may receive multiple Layer-2 flows (also referred to as Attachment Circuits (ACs)). In a typical network supporting Pseudowires, each AC is mapped to a Pseudowire. Ingress packets received mapped to a specific Pseudowire are labeled with an identifier associated with this Pseudowire, and are switched via the Pseudowire. A physical link may support one or more Pseudowires. Ideally, the data flow in a Pseudowire should be protected. In other words, if an active Pseudowire fails, the data flow should be redirected to an alternative Pseudowire to avoid data loss.

Pseudowires can operate over many physical media types. However, existing Pseudowire systems typically provide no protection or very limited protection. For example, there is usually no data protection for Pseudowires on different physical media types, since most network protection schemes, such as APS for SONET, Link Aggregation for Ethernet, do not apply over multiple physical media types.

Some MPLS devices implement schemes such as MPLS Fast Reroute to provide limited data protection. These existing schemes, however, often do not provide adequate protection. Take the following scenario as an example: between two provider edges (PEs), a first tunnel comprising multiple Pseudowires is protected by a second tunnel. Due to network topology constraints, the two tunnels may have different bandwidth. This is a possible scenario in an MPLS Fast Reroute operation. In this example, the second tunnel may have lower bandwidth than that of the first one. If the first tunnel should fail, the amount of data that needs to be redirected through the second tunnel may exceed the capacity of the second tunnel. Furthermore, existing protocols typically do not provide a way of determining which data gets priority. Thus, certain mission critical data may be dropped while other less critical data may pass through.

It would be desirable to have a way to provide better Pseudowire protection and to have more control during switchover. It would also be desirable if the protection scheme could be implemented without significant changes to existing protocols and devices.

BRIEF DESCRIPTION OF THE DRAWINGS

Various embodiments of the invention are disclosed in the following detailed description and the accompanying drawings.

FIGS. **1A** and **1B** are block diagrams illustrating an embodiment of a single-hop Pseudowire system and an embodiment of a multi-hop Pseudowire system, respectively.

FIG. **2** is a flowchart illustrating an embodiment of a process of providing data protection using Pseudowires.

FIG. **3A** is a flowchart illustrating another embodiment of a process of providing data protection using Pseudowires.

FIG. **3B** is a flowchart illustrating how the Pseudowire is used, according to some embodiments.

FIG. **4** is a data structure diagram illustrating an embodiment of a Pseudowire protection configuration parameter that specifies several protection-related properties of the Pseudowire.

FIG. **5** is a flowchart illustrating an example process of using the priorities during switchover.

FIG. **6** is a diagram illustrating am example in which preemption takes place during a switchover operation.

DETAILED DESCRIPTION

The invention can be implemented in numerous ways, including as a process, an apparatus, a system, a composition of matter, a computer readable medium such as a computer readable storage medium or a computer network wherein program instructions are sent over optical or electronic communication links. In this specification, these implementations, or any other form that the invention may take, may be referred to as techniques. A component such as a processor or a memory described as being configured to perform a task includes both a general component that is temporarily configured to perform the task at a given time or a specific component that is manufactured to perform the task. In general, the order of the steps of disclosed processes may be altered within the scope of the invention.

A detailed description of one or more embodiments of the invention is provided below along with accompanying figures that illustrate the principles of the invention. The invention is described in connection with such embodiments, but the invention is not limited to any embodiment. The scope of the invention is limited only by the claims and the invention encompasses numerous alternatives, modifications and equivalents. Numerous specific details are set forth in the following description in order to provide a thorough understanding of the invention. These details are provided for the purpose of example and the invention may be practiced according to the claims without some or all of these specific details. For the purpose of clarity, technical material that is known in the technical fields related to the invention has not been described in detail so that the invention is not unnecessarily obscured.

Providing protection to network traffic using one or more Pseudowires is disclosed. In some embodiments, a Pseudowire protection configuration parameter is sent to a destination node. A Pseudowire configuration acknowledgment from the destination node is received. If a Pseudowire is allowed to be established according to the Pseudowire configuration acknowledgment, it is established based at least in part on the Pseudowire protection configuration parameter. In embodiments where the Pseudowire is established as a standby

Pseudowire configured to protect one or more primary Pseudowires, in the event that a primary Pseudowire fails to transfer network traffic for reasons such as network congestion, equipment failure, etc., network traffic that is originally designated to be transferred on the primary Pseudowire(s) is switched from the primary Pseudowire(s) to the standby Pseudowire.

The protection technique is applicable to both single-hop and multi-hop systems. FIGS. **1A** and **1B** are block diagrams illustrating an embodiment of a single-hop Pseudowire system and an embodiment of a multi-hop Pseudowire system, respectively. Configuring and switching the Pseudowire will be discussed in more detail below.

In the example shown in FIG. **1A**, system **100** is a single-hop system where the nodes in the system all belong to the same carrier network. Within each carrier network, all network nodes and facility are under a common administrative control. A service provider company may own multiple carrier networks in different regions. As used herein, a node refers to a networked device. In this case, the nodes in the system are provider edges (PEs) A, B, C, and D, which all belong to the same carrier network. Ingress data received by attachment circuits **112** of PE A designated for PE B may be sent via a label switched path (LSP) through PEs A, C, and B, or an LSP through PEs A, D, and B. The first LSP comprises Pseudowires **102**, **104** and **106**, and the second LSP comprises Pseudowires **108** and **110**. In this example, the Pseudowire connections between PEs are established using the Label Distribution Protocol (LDP). The connections are based on LDP sessions. Each LDP session is to connect two local or remote nodes. There may be multiple paths interconnecting any two nodes in the network. Thus, for each LDP session, there may be multiple LDP Hello Adjacencies, one LDP Hello Adjacency per path. For purposes of example, throughout this specification, LDP is used as the communication protocol between nodes. Other appropriate protocols may also be used.

In the example shown in FIG. **1B**, system **150** is a multi-hop system since it includes multiple carrier networks. Carrier networks **1**-**6** form autonomous systems **1**-**6**, respectively. Each autonomous system includes one or more networks that are controlled by a carrier. For purposes of illustration, three Pseudowires are shown in this example to transfer data between PE **1A** and PE **3B**: a first Pseudowire comprising a path via autonomous systems **1**, **2**, and **3**, a second Pseudowire comprising a path via autonomous systems **1**, **6**, and **3**, and a third Pseudowire comprising a path via autonomous systems **1**, **4**, **5**, and **3**. Other Pseudowire formations are possible. At the source node PE **1A**, data packets to be sent via a particular Pseudowire are labeled with an identifier associated with the Pseudowire, forwarded on to the next provider edge on one Pseudowire segment, and forwarded again if necessary until the packets reach the destination node **3B**.

FIG. **2** is a flowchart illustrating an embodiment of a process of providing data protection using Pseudowires. Process **200** may be implemented on a source node such as A or 1 A of systems **100** and **150**, or on an independent management agent that communicates with the source node. For purposes of illustration, the process is shown as implemented on a source node in the following example. The process initializes when a connection session is established between the source node and the destination node (**202**). A Pseudowire protection configuration parameter for configuring a Pseudowire based on the connection session is sent (**204**). The Pseudowire protection configuration parameter includes one or more fields that specify certain protection properties associated with the Pseudowire. It may be sent to the destination node or a management agent that communicates with the destination node. Details of the configuration parameter will be discussed further below.

Once the destination node (or its associated management agent) receives the Pseudowire protection configuration parameter, it determines whether it will accept the Pseudowire protection configuration and allow a standby Pseudowire to be established. Depending on implementation, the destination node determines whether to accept the protection configuration based on factors such as traffic condition, number of existing Pseudowires, priority information, etc. The destination node may reject the protection request for a number of reasons. For example, the destination node does not support Pseudowire protection mechanism as described here. If a standby Pseudowire may be established, the destination node accepts it and configures the Pseudowire based at least in part on the configuration parameters. In some embodiments, the destination node adds the Pseudowire to a table of Pseudowires. A corresponding Pseudowire configuration acknowledgment is generated, indicating whether the destination node has accepted the Pseudowire configuration. The Pseudowire configuration acknowledgment is sent to the source node. In some embodiments, as a part of the LDP process, a MPLS label for the data packets traversing through the standby Pseudowire is assigned.

At the source node, once the Pseudowire configuration acknowledgment is received (**206**), it is examined to determine whether the Pseudowire configuration has been accepted (**208**). If, according to the Pseudowire configuration acknowledgment, the Pseudowire configuration has been accepted by the destination, a standby Pseudowire is established based at least in part on the Pseudowire protection configuration parameter and may be used as such (**210**). If, however, the Pseudowire configuration has not been accepted, the process performs appropriate exception handling, such as re-sending the Pseudowire protection configuration parameter (**212**).

FIG. **3A** is a flowchart illustrating another embodiment of a process of providing data protection using Pseudowires. Process **300** may be implemented on a PE, on an independent management agent, or the like. For purposes of illustration, in the following example, the process is initiated and carried out on a PE source node.

Process **300** begins with the initialization of an LDP session (**302**). According to the negotiation scheme based on LDP, the source node exchanges messages with the destination node and establishes an LDP Hello Adjacency (**304**). A Pseudowire setup request that includes a Pseudowire protection configuration parameter is sent to the destination node (or its associated management agent), requesting that a standby Pseudowire be established over the LDP Hello Adjacency (**306**). In some embodiments, multiple LDP Hello Adjacencies are available for Pseudowire setup, thus multiple setup requests are sent, and the destination node processes the requests and maps Pseudowires to appropriate LDP Hello Adjacencies. In some embodiments, the source node dynamically determines which LDP Hello Adjacency among the available connections is to be configured as a standby Pseudowire, and directs its setup request accordingly. The dynamic determination may be based on, among other things, bandwidth availability on the adjacency path.

In some embodiments, the request is sent as a LDP Label Mapping Message. The configuration parameter is used to configure various properties of the Pseudowire, including protection type, protection scheme, priority, etc. Further details of the configuration parameters are discussed below. In some embodiments, multiple LDP Hello Adjacencies are established and the source node sends multiple Pseudowire setup requests to configure Pseudowires over these LDP Hello Adjacencies.

In this example, upon receiving a Pseudowire setup request, the destination node maps the request to the appropriate LDP Hello Adjacency. If the mapping is successful, the Pseudowire is established. Sometimes, however, the mapping and consequently the Pseudowire setup may fail for reasons such as network congestion, resource limitation, equipment failure, etc. The destination node sends a Pseudowire configuration acknowledgment to the source node. In this example, the Pseudowire configuration acknowledgment is an LDP acknowledgement indicating whether a particular Pseudowire has been successfully established. Once the source node receives the acknowledgement (**308**), it determines whether the configuration has been accepted by the destination (**310**). If the configuration has been accepted, a standby Pseudowire is successfully established based at least in part on the Pseudowire protection configuration parameter, and the source and destination nodes can start using the standby Pseudowire to protect other Pseudowires (**312**). If, however, the acknowledgment indicates that the configuration has not been accepted and a Pseudowire has not been successfully established, appropriate exception handling measures such as resending the Pseudowire protection configuration parameter are taken (**314**).

Process **300** is applicable to both single-hop and multi-hop systems. In a single-hop system, the source node and the destination node correspond to a source PE and a destination PE on the network and the process is used to configure a standby Pseudowire between the PEs. In a multi-hop system, the process may be repeated by the PEs on various carrier networks to establish Pseudowire segments. For example, in system **150** of FIG. 1B, PE **1A** can use process **300** to establish a Pseudowire segment with PE **6A**, and PE **6A** can use the same process to establish-a Pseudowire segment with PE **6B**, which can use the same process to establish a Pseudowire segment with PE **3B**.

FIG. **3B** is a flowchart illustrating how the Pseudowire is used, according to some embodiments. Process **350** may be implemented on the source node, the destination node, or both. In this example, the designation of the Pseudowire is first determined (**352**). The designation may be configured by a system administrator, in an Pseudowire configuration process, or any other appropriate means. If the Pseudowire is designated as a primary Pseudowire, it is configured to carry network traffic (**354**). In the event that a primary Pseudowire fails (**356**), the nodes associated with the Pseudowire will attempt to switch the traffic over to the standby Pseudowire by sending a switchover request to the Pseudowire (**358**). As will be shown in more detail below, in some embodiments, whether the traffic on the primary Pseudowire can preempt the traffic on the standby Pseudowire and be switched over depends on priority configuration of the Pseudowires.

If it is designated as a standby Pseudowire, it is enters into standby mode to provide protection to one or more primary Pseudowires (**360**). In some embodiments, the standby Pseudowire carries network traffic during normal operation. It is ready to take over traffic from the primary Pseudowire if necessary. If a switchover request is received from a primary Pseudowire (**362**), traffic on the primary Pseudowire is switched over to the standby Pseudowire. In some embodiments, the switchover only occurs if the priority comparison of the primary and standby Pseudowires indicates the switchover is allowed.

Optionally, during the operation, if a Pseudowire is no longer needed, the source node can send a withdraw request

over the Pseudowire and the destination node disassociates the Pseudowire with the LDP Hello Adjacency to break the Pseudowire connection.

FIG. **4** is a data structure diagram illustrating an embodiment of a Pseudowire protection configuration parameter that specifies several protection-related properties of the Pseudowire. In this example, Pseudowire protection configuration parameter **400** includes four fields: protection scheme, protection type, domain type, and priority. A field may have one or more subfields. For example, the priority field is shown to include a holding priority and a setup priority. One or more of the fields and/or subfields may be used in various embodiments. Other appropriate fields may also be implemented. In the example shown, the fields are numerical values that map to appropriate property values.

In some embodiments, one of the following Pseudowire protection schemes is used to set up the Pseudowires: 1+1, 1:1, 1:N or M:N. The protection scheme field is used to indicate which protection scheme is used in the system setup. A specific protection scheme corresponds to a field value. For example, 1+1 maps to 0, 1:1 maps to 1, and so on. In a system implementing a 1+1 protection scheme, the same traffic is sent over two parallel Pseudowires and the receiver selects one traffic stream at a time. In a system implementing a 1:1 protection scheme, one Pseudowire is used is used to protect another Pseudowire. Similarly, in a 1:N system (e.g. MPLS Facility Backup), one Pseudowire is used to protect N other Pseudowires, and in a M:N system M Pseudowires are used to protect N other Pseudowires.

The protection type field is used to configure the standby mode of the Pseudowire. In some embodiments, cold, warm, and hot standby modes are supported. Other appropriate standby modes may be implemented in other embodiments. In some embodiments, in cold standby mode configuration, once network failure on a Pseudowire carrying network traffic is detected, a standby Pseudowire is selected from the remaining functional Pseudowires, and traffic is redirected to the standby Pseudowire. In some embodiments with warm standby mode configuration, one or more standby Pseudowires are established before any network failure has occurred. These standby Pseudowires, however, are not maintained or used to transport data until a network failure is detected. Upon failure detection, the source or destination nodes will modify the data-plane and switch data traffic over to the standby Pseudowire(s). In some embodiments with hot standby mode configuration, one or more standby Pseudowires are pre-established and maintained at both control-plane and data-plane, so that once a network failure is detected, data traffic is directly switched over to the standby Pseudowire(s).

The domain type field indicates whether the Pseudowire is configured in a single-hop environment where all the nodes of the Pseudowire belong to the same carrier network, or a multi-hop environment where the Pseudowire includes nodes on several carrier networks. This is because the intermediate may process single-hop and multi-hop Pseudowire differently.

The priority field indicates the preference level of a Pseudowire in preempting other Pseudowires during switchover. In the event of a network failure, the edge nodes will preferentially provide protection according to the priority setting of the Pseudowires. In a situation where network resources (such as bandwidth) are limited, data sent on a higher priority Pseudowire is more likely to be protected than data sent on a lower priority Pseudowire. In some embodiments, the priority field includes two subfields: a holding priority and a setup priority. The holding priority indicates the relative priority of a currently active Pseudowire with respect to other Pseudowires when the latter attempt to preempt the former's use of the data link. Stated another way, it determines how easily a currently active Pseudowire gives up its hold on a data link upon request. The setup priority indicates the relative priority of a Pseudowire during the setup process.

FIG. **5** is a flowchart illustrating an example process of using the priorities during switchover. Process **500** may be implemented on an edge node, an independent management agent, or the like. In this example, process **500** initiates when a network failure has been detected (**502**). It is determined whether preemption is required (**504**). Preemption is required when the failed link carries more Pseudowire traffic than the available bandwidth on the standby link. If preemption is not required, the Pseudowire(s) may directly switchover (**506**). If, however, preemption is required, the setup priorities of the Pseudowires on the failed link are compared and the Pseudowire with the highest setup priority is selected (**508**). The setup priority of the selected Pseudowire is compared to the holding priority of the standby Pseudowire (**510**). If the setup priority is greater than the holding priority, traffic on the selected Pseudowire is switched over to the standby Pseudowire (**506**). If, however, the setup priority is no greater than the holding priority, no switchover takes place and the standby Pseudowire continues to transfer its own data and the data on the failed Pseudowires is lost (**514**).

FIG. **6** is a diagram illustrating am example in which preemption takes place during a switchover operation. In this example, Pseudowires **600**, **602** and **604** are active, primary Pseudowires carrying traffic. Pseudowire **604** is used as the standby Pseudowire. Pseudowire **600** has a holding priority and a setup priority of 10 and 11, respectively, Pseudowire **602** has priorities of 11 and 12, and Pseudowire **604** has priorities of 9 and 9. Thus, if the link on which Pseudowires **600** and **602** operate fails, the nodes will initiate switchover using Pseudowire **604**. A comparison of the setup priority of Pseudowires **600** and **602** indicates that Pseudowire **602** has a higher setup priority, thus **602** is given preference in the switchover. The setup priority of Pseudowire **602** is compared with the holding priority of Pseudowire **604**. Since **602**'s setup priority is greater than **604**'s holding priority, data on **602** preempts data on **604** and takes over the link.

Providing protection to network traffic using one or more Pseudowires has been disclosed. Pseudowire protection improves the reliability of Pseudowire services. Pseudowires are better controlled by appropriately configuring the properties of Pseudowires and without requiring significant changes to existing protocols and devices.

Although the foregoing embodiments have been described in some detail for purposes of clarity of understanding, the invention is not limited to the details provided. There are many alternative ways of implementing the invention. The disclosed embodiments are illustrative and not restrictive.

What is claimed is:

1. A method of providing protection to network traffic, comprising:
 - sending a Pseudowire protection configuration parameter for configuring a standby Pseudowire between a source node and a destination node, the Pseudowire protection configuration parameter indicating a protection property associated with the standby Pseudowire, the protection property including a priority for the standby Pseudowire;
 - receiving a Pseudowire configuration acknowledgement indicating whether the Pseudowire protection configuration parameter has been accepted by the destination node;

 - accepting the Pseudowire protection configuration parameter by the destination node;
 - using the standby Pseudowire that is configured based at least in part on the Pseudowire protection configuration parameter; and
 - determining whether to preempt existing traffic on the standby Pseudowire, wherein the determination is based, at least in part, on the priority for the standby Pseudowire.

2. A method as recited in claim **1**, wherein the standby Pseudowire is configured to provide protection to at least one primary Pseudowire.

3. A method as recited in claim **1**, wherein the standby Pseudowire is configured to provide protection to at least one primary Pseudowire, and in the event that the primary Pseudowire fails to transfer network traffic, switching network traffic from at least one of said at least one primary Pseudowire to the standby Pseudowire.

4. A method as recited in claim **1**, wherein the standby Pseudowire is dynamically selected from a plurality of connections.

5. A method as recited in claim **1**, wherein the protection property further includes at least one of a domain type, a protection type or a protection scheme.

6. A method as recited in claim **1**, wherein the Pseudowire protection configuration parameter is established using the Label Distribution Protocol (LDP).

7. A method as recited in claim **5**, wherein the domain type indicates whether the standby Pseudowire is configured in a single-hop environment where the standby Pseudowire includes a plurality of nodes coupled to a same carrier network, or a multi-hop environment where the standby Pseudowire includes a plurality of nodes coupled to several carrier networks.

8. A method as recited in claim **5**, wherein the protection scheme indicates at least one of the following:
 - a 1+1 protection scheme, wherein the same traffic is sent over two Pseudowires;
 - a 1:1 protection scheme, wherein one standby Pseudowire is used to protect another Pseudowire;
 - a 1:N protection scheme, wherein one standby Pseudowire is used to protect N other Pseudowires; or
 - an M:N protection scheme, wherein M standby Pseudowires are used to protect N other Pseudowires.

9. A system for providing protection to network traffic, comprising:
 - a processor configured to:
   - send a Pseudowire protection configuration parameter for configuring a standby Pseudowire between a source node and a destination node, the Pseudowire protection configuration parameter indicating a protection property associated with the standby Pseudowire, the protection property including a priority for the standby Pseudowire;
   - receive a Pseudowire configuration acknowledgement indicating whether the Pseudowire protection configuration parameter has been accepted by the destination node;
   - accept the Pseudowire protection configuration parameter by the destination node;
   - use the standby Pseudowire that is configured based at least in part on the Pseudowire protection configuration parameter; and
   - determine whether to preempt existing traffic on the standby Pseudowire, wherein the determination is based, at least in part, on the priority for the standby Pseudowire.

10. A system as recited in claim **9**, wherein the standby Pseudowire is configured to provide protection to at least one primary Pseudowire.

11. A system as recited in claim **9**, wherein the protection property further includes at least one of a domain type, a protection type or a protection scheme.

12. A system as recited in claim **11**, wherein the domain type indicates whether the standby Pseudowire is configured in a single-hop environment where the standby Pseudowire includes a plurality of nodes coupled to a same carrier network, or a multi-hop environment where the standby Pseudowire includes a plurality of nodes coupled to several carrier networks.

13. A system as recited in claim **11**, wherein the protection scheme indicates at least one of the following:

a 1+1 protection scheme, wherein the same traffic is sent over two Pseudowires;

a 1:1 protection scheme, wherein one standby Pseudowire is used to protect another Pseudowire;

a 1:N protection scheme, wherein one standby Pseudowire is used to protect N other Pseudowires; or

an M:N protection scheme, wherein M standby Pseudowires are used to protect N other Pseudowires.

14. A computer program product for configuring a Pseudowire between a source node and a destination node, the computer program product being embodied in a computer readable storage medium and comprising computer instructions for:

sending a Pseudowire protection configuration parameter for configuring a standby Pseudowire between a source node and a destination node, the Pseudowire protection configuration parameter indicating a protection property associated with the standby Pseudowire, the protection property including a priority for the standby Pseudowire;

receiving a Pseudowire configuration acknowledgement indicating whether the Pseudowire protection configuration parameter has been accepted by the destination node;

accept the Pseudowire protection configuration parameter by the destination node;

using the standby Pseudowire that is configured based at least in part on the Pseudowire protection configuration parameter; and

determining whether to preempt existing traffic on the standby Pseudowire, wherein the determination is based, at least in part, on the priority for the standby Pseudowire.

15. A computer program product as recited in claim **14**, wherein the protection property further includes at least one of a domain type, a protection type or a protection scheme.

16. A computer product as recited in claim **15**, wherein the domain type indicates whether the standby Pseudowire is configured in a single-hop environment where the standby Pseudowire includes a plurality of nodes coupled to a same carrier network, or a multi-hop environment where the standby Pseudowire includes a plurality of nodes coupled to several carrier networks.

17. A computer product as recited in claim **15**, wherein the protection scheme indicates at least one of the following:

a 1+1 protection scheme, wherein the same traffic is sent over two Pseudowires;

a 1:1 protection scheme, wherein one standby Pseudowire is used to protect another Pseudowire;

a 1:N protection scheme, wherein one standby Pseudowire is used to protect N other Pseudowires; or

an M:N protection scheme, wherein M standby Pseudowires are used to protect N other Pseudowires.

* * * * *